Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Konekcionistické modelovanie učenia sa analógií

Diplomová práca

2014 Bc. Peter Gergeľ

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Konekcionistické modelovanie učenia sa analógií

Diplomová práca

Študijný program: Aplikovaná informatika
Študijný odbor: 2511 Aplikovaná informatika
Školiace pracovisko: Katedra aplikovanej informatiky
Školiteľ: doc. Ing. Igor Farkaš, PhD.

Bratislava, 2014 Bc. Peter Gergeľ

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

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

Názov: Konekcionistické modelovanie učenia sa analógií / *Connectionist modeling of analogy learning*

Cieľ: 1. Urobte stručný prehľad výpočtových modelov vytvárania analógií (symbolových, konekcionistických, hybridných) a posúďte ich vlastnosti.
2. Vyberte si konekcionistický model, a navrhnite jeho vylepšenie s cieľom zvýšiť úspešnosť modelu pri učení sa analógií.
3. Pomocou počítačových simulácií vyhodnoťte správanie navrhnutého modelu vo zvolenej úlohe.

Literatúra: French, R.M. (2002). The computational modeling of analogy-making. Trends in Cognitive Sciences, 6(5), 200-205.
Blank, D. S. (1997). Learning to see analogies: A connectionist exploration. Dizertačná práca, Indiana University.

Anotácia: Vytváranie analógií je dôležitou kognitívnou schopnosťou, ktorá zahŕňa schopnosť človeka vysvetľovať nové koncepty pomocou známych, schopnosť zdôrazniť niektoré aspekty situácií, generalizovať, charakterizovať situácie, vysvetliť alebo opísať nové fenomény, môže poslúžiť ako základ na to, ako konať pri nových okolnostiach, porozumieť rôznym formám humoru, atď. V tomto kontexte je dôležitosť kognitívneho modelovania vytvárania analógií zrejmá.

Kľúčové slová: učenie sa analógií, podobnosť, výpočtové modely, kognícia

Vedúci: doc. Ing. Igor Farkaš, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. PhDr. Ján Rybár, PhD.

Dátum zadania: 30.11.2012

Dátum schválenia: 30.11.2012

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

Gergeľ
.....
študent

.....
vedúci práce

Čestne prehlasujem, že som túto prácu vypracoval sám za pomoci uvedenej literatúry a pod starostlivým dohľadom školiteľa.

. . . . . .

Poďakovanie

Týmto by som sa chcel poďakovať môjmu školiteľovi doc. Ing. Igorovi Farkašovi, PhD. za jeho podporu, čas, priateľský prístup a neoceniteľné rady a pripomienky, ktorými ma viedol správnym smerom pri tvorbe tejto práce.

Ďalej by som sa chcel poďakovať RNDr. Martinovi Takáčovi, PhD. za skvelý postreh, a mojim rodičom a priateľom za podporu.

Abstrakt

Analogické usudzovanie, popri induktívnom, deduktívnom, či abduktívnom usudzovaní, patrí medzi základné mechanizmy, ktorými človek spoznáva svet, učí sa, či rieši problémy. Modelovanie tejto schopnosti pomocou počítačových simulácií je dôležité, pretože môže ponúkať vysvetlenia, ako tento mechanizmus funguje. V práci sme sa zamerali na oblasť učenia sa analógií typu *časť–celok*, pri ktorých sú analogické objekty medzi dvoma scénami tie, ktoré sú si nejakým spôsobom podobné. Pri modelovaní sme využili konekcionistickú paradigmu na báze umelých neurónových sietí. Zamerali sme sa na problém geometrických analógií, pričom vychádzame z modelu *Analogator*, ktorý mal obmedzenia brániace vo využití celého jeho potenciálu a zároveň zbytočne zvyšovali časovú a pamäťovú náročnosť. V práci prinášame metódy, ktorými odstraňujeme tieto obmedzenia, vďaka čomu potenciál modelu narástol.

KĽÚČOVÉ SLOVÁ: učenie sa analógií, podobnosť, výpočtové modely, kognícia, Analogator, jednoduchá rekurentná sieť

Abstract

Analogical reasoning, along with inductive, deductive or abductive reasoning, belongs to the fundamental human mechanisms for the environment exploration, learning or problem solution. Modeling this ability using computer simulations is important, as it might offer an explanation how this mechanism works. In the work, we focus on the *part–whole* analogies where the analogical objects between two scenes show a mutual resemblance. For simulations we use a connectionist paradigm based on the artificial neural networks. We focus on the problem of geometrical analogies while we proceed from the model *Analogator* which had limitations hindering the usage of its full potential and at the same time, these limitations increased the time and memory complexity. We offer methods for removing these limitations due to which the potential of the model was enhanced.

KEYWORDS: analogy learning, similarity, computational models, cognition, Analogator, simple recurrent network

Obsah

Úvod

Analogické uvažovanie je fundamentálna kognitívna schopnosť ľudí, ktorou sa líšia od väčšiny zvierat.[1] Tento mechanizmus umožňuje vysvetľovať nové koncepty pomocou známych, zdôrazňovať niektoré aspekty situácií, generalizovať, charakterizovať situácie, vysvetliť alebo opísať nové fenomény, môže poslúžiť ako základ na to, ako konať pri nových okolnostiach, porozumieť rôznym formám humoru, atď. Koncept učenia a vytvárania si analógií u ľudí nie je stále plne pochopený, preto je kognitívne modelovanie dôležité, lebo môže ponúkať hypotézy a vysvetlenia.

Problém, ktorý sme si zvolili, je problém geometrických analógií, pri ktorom je úlohou určiť geometrický objekt v jednej scéne, ktorý je analogický s vopred určeným geometrickým objektom v inej scéne. Tento problém spadá pod doménu analógií typu *časť–celok*, pri ktorých sú analogické objekty medzi dvoma scénami tie, ktoré zohrávajú vo svojich scénach rovnakú úlohu, majú rovnakú, resp. podobnú, štruktúru, či percepčne sa odlišujú tým istým spôsobom.

Konekcionistické riešenie týchto analógií typu *časť–celok* navrhol Blank (1997) vo svojej dizertačnej práci, ktorou sme sa inšpirovali. Jeho model *Analogator* je založený na jednoduchej rekurentnej sieti, ktorá sa učí analogicky uvažovať tak, že vidí veľa príkladov analógií. To je podstatne iný prístup oproti ostatným modelom, ktoré sú väčšinou priamo stavané na analogické uvažovanie a zvyčajne im chýba schopnosť učenia. *Analogator* mal obmedzenia, ktoré mu bránili vo využitiu celého jeho potenciálu a zároveň zbytočne zvyšovali časovú a pamäťovú náročnosť. My sme navrhli viacero spôsobov, ako tieto obmedzenia odstrániť, a zároveň sme preskúmali možnosti zvýšenia potenciálu modelu.

Práca je členená do štyroch kapitol. V prvej kapitole sa venujeme analogickému uvažovaniu u ľudí, ukážeme ako možno využiť analógie na riešenie problémov a poskytujeme stručný prehľad výpočtových modelov, ktoré boli navrhnuté na vytváranie analógií. V druhej kapitole opisujeme problém geometrických analógií s niekoľkými variantami, ktoré

[1] Človek však nie je jediný, kto ma túto schopnosť. Experimenty na zvieratách ukázali, že aj šimpanzy (*Pan troglodytes*) sú toho schopné (Thompson a spol., 1997).

sme preskúmali. V tretej kapitole špecifikujeme Blankov model *Analogator* a zároveň v nej navrhujeme riešenia obmedzení modelu a možnosti zvýšenia výkonu. Nakoniec vo štvrtej kapitole vyhodnotíme správanie základného modelu, spolu s našimi modelmi, na probléme geometrických analógií.

Kapitola 1

Analógie – ľudia a stroje

1.1 Analogické uvažovanie u ľudí

Vytvoriť analógiu znamená vidieť nejaký objekt, alebo situáciu, v jednom kontexte rovnako ako iný objekt, alebo situáciu, v druhom kontexte. Hall (1989) vysvetľuje tento proces ako zobrazenie medzi dvoma doménami nazvanými *zdroj*, ktorý predstavuje známu situáciu, a *cieľ*, ktorý predstavuje novú, neznámu situáciu. Medzi týmito štruktúrami sa objekty, ktoré majú rovnaký význam vrámci domény, zobrazia na seba. Tento proces pozostáva z týchto štyroch krokov:

1. Identifikácia zdroja.
2. Určenie miery podobnosti.
3. Prenos znalostí zo zdroja na cieľ.
4. Konsolidácia.

Človek si pri vytváraní analógie potrebuje vybaviť nejakú situáciu z minulosti, medzi ktorou vidí podobnosť s aktuálnou situáciou (*identifikácia zdroja*). Premyslí si, či podobnosť medzi známou a novou situáciou je vysoká (*určenie miery podobnosti*) a ak áno, pokúsi sa aplikovať poznatky zo známej situácie na novú situáciu (*prenos znalostí zo zdroja na cieľ*). Výsledok aplikovania starých poznatkov v novej situácii si zapamätá (*konsolidácia*).

Existuje niekoľko foriem analogického uvažovania, pričom rôzne formy majú rôzny historický vývoj. Za klasickú formu analógie, ktorá sa datuje do doby starovekého Grécka, sa považuje *propozičná analógia*, ktorá má tvar *A ku B je ako C ku D*. Patria sem *synonymické analógie* (profesor k učiteľovi je ako študent ku žiakovi), *antonymické analógie* (biela k čiernej je ako teplá k zimnej), alebo analógie typu *časť–celok* (strom ku lesu je

ako katedra ku fakulte). Za ďalšiu možnú formu možno považovať *metafory*, pri ktorých sa prenáša význam na základe podobností niektorých znakov (Jožo je veľký ako slon) a v neposlednom rade forma analógie, pri ktorej sa nešpecifikuje povaha analógie (divadlo je analogické televízii).

V prehľadovej práci o vývine analógii u ľudí, Goswami (1991) uvádza, že schopnosť analogicky uvažovať dokážu už dvojročné deti. V experimentoch tieto deti dokázali riešiť klasickú formu analógie *A ku B je ako C ku D* a zároveň dokázali riešiť *cieľové* problémy potom čo dostali *zdrojové* problémy aj s ich riešeniami.

Mithen (1996) vo svojom diele hovorí, že v minulosti počas evolučného vývoje ľudskej mysle, človek mal špecializované typy *inteligencie* ako sú napr. *všeobecná inteligencia*, *emočná inteligencia*, či *sociálna inteligencia*. Tieto typy inteligencie boli nezávislé a oddelené, žiadna z nich nemala prístup k inej inteligencii. Ako sa človek evolučne vyvíjal, nastala integrácia týchto inteligencií, vďaka čomu sa zrodilo aj analogické uvažovanie.

1.1.1 Problém pevnosti a problém ožiarenia

Vďaka analogickému uvažovaniu je možné riešiť problémy tak, že riešenie jedného problému, ktoré je k dispozícii, sa nejakým spôsobom pretransformuje na riešenie druhého problému. Analógia medzi problémom pevnosti (angl. fortress problem) a problémom ožiarenia (angl. radiation problem) je ukážkovým príkladom tejto schopnosti. Duncker a Lees (1945) vymysleli vo svojej knihe takýto problém:

> Ako doktor máte vyliečiť pacienta, ktorý ma zhubný, neoperovateľný nádor. Pokiaľ nádor nebude odstránený, pacient zomrie. K dispozícii máte špeciálny lúč, ktorým nádor môžete zničiť. Ak použijete lúč pri vysokej intenzite, nádor zničíte, no zároveň zničíte aj zdravé tkanivo, ktoré stálo v ceste. Ak by sa použil lúč pri nízkej intenzite, zdravé tkanivo by zostalo nepoškodené, ale nádor by to nezničilo. Akú procedúru musíte zvoliť, aby ste pomocou lúča nádor zničili, ale zároveň ste nepoškodili zdravé tkanivo?

Tento problém použili Gick a Holyoak (1980, 1983) vo svojich dvoch experimentoch, kde ho prezentovali účastníkom a zaujímalo ich koľko účastníkov dokáže nájsť riešenie. Väčšina z nich to nedokázala a tak im zadali ďalší problém aj s jeho riešením:

> Generál chce dobiť nepriateľskú pevnosť. Zhromaždil obrovskú armádu a plánuje hromadný útok, lenže zistí, že všetky cesty vedúce k pevnosti sú zamínované. Cesty sú navrhnuté tak, že malá skupina ľudí dokáže nimi bezpečne

> prejsť, ale každá väčšia skupina by míny aktivovala. Generál vymyslel plán, že rozdelí svojich vojakov do malých skupín tak, aby každá skupina dokázala bezpečne prejsť cestou, a odchod každej skupiny po svojej ceste naplánuje tak, aby na konci dorazili všetky skupiny k pevnosti súčasne.

Ako sme uvádzali v úvode, proces vytvorenia analógie prebieha medzi doménami *zdroj* a *cieľ*. *Zdroj* je v tomto prípade problém pevnosti a *cieľ* je problém ožiarenia. Tým, že sa účastníkom poskytli obe problémy, sa očakávalo, že účastníci si všimnú podobnosť a dokážu problém ožiarenia vyriešiť analogicky. Správne riešenie problému ožiarenia teda bolo použiť viacero lúčov s nízkou intenzitou (rozdelenie armády na malé skupiny) a vystreliť ich súčasne z viacerých uhlov na nádor (poslanie armády rôznymi cestami, aby na konci dorazili všetci súčasne). Prvý krok vytvorenia analógie, *identifikácia zdroja*, vykonali experimentátori a účastníci mali za úlohu vykonať ďalšie kroky.

Výrazný pokrok pri riešení pôvodného problému, problému ožiarenia, nenastal. Predtým než sa účastníci dozvedeli o probléme pevnosti s jeho riešením, iba 10% vedelo vymyslieť správne riešenie. Zo zvyšných 90%, ktorým bol problém pevnosti podaný, len 30% vedelo použiť riešenie problému na vyriešenie problému ožiarenia a 75%–80% to vedelo vyriešiť po poskytnutí nápovedy. Z toho vyplýva, že len približne štvrtina zúčastnených si dokázala vytvoriť analógiu medzi problém pevnosti a problémom ožiarenia.

Dôvody zlyhania pri vytváraní analógie

Existuje viacero dôvodov, prečo ľudia zlyhávajú pri vytváraní analógie. Jeden z nich môže byť ten, že niektoré všeobecné poznatky môžu brániť videniu analógie. Účastníci si napríklad mohli predstavovať zariadenie generujúce lúč ako zariadenie s obrovskou veľkosťou, ktoré sa nezmestí do miestnosti súčasne s ďalšími takýmito zariadeniami. Alebo si nevedeli predstaviť to, ako doktor dokáže pracovať naraz s viacerými lúčmi, ak má len dve ruky.

Ďalší dôvod môže byť ten, že účastníci si môžu pridávať do zadania problému vlastné obmedzenia, ktorá sa tam v skutočnosti nenachádzajú. Napríklad vetu: „K dispozícií máte špeciálny lúč, ktorým nádor môžete zničiť.” mohli chápať tak, že existuje práve jeden lúč, aj keď to vo vete nie je napísané.

V neposlednom rade môže robiť problémy aj to, že pri vytváraní analógie sa ľudia snažia zobraziť *všetky* objekty zdrojovej domény na objekty cieľovej domény. V rámci vyššie uvedenej analógie sa vojaci zobrazujú na lúč, množstvo vojakov sa zobrazuje na intenzitu lúča, pevnosť sa zobrazuje na nádor. Účastníci sa možno snažili zobraziť míny, stromy, zbrane vojakov na nejaké objekty z domény ožiarenia, čo sa im nemuselo podariť

a celá analógia kvôli tomu zlyhala.

1.2 Analogické uvažovanie u strojov

French (2002) rozdeľuje výpočtové modely, ktoré sa podieľajú na vytváraní analógií, na symbolové modely, konekcionistické modely a hybridné modely.

Symbolové modely

Symbolové modely používajú metódy klasickej, symbolovej umelej inteligencie. Dominantnú úlohu tu hrajú symboly, logika, plánovanie, prehľadávanie a pod.

Konekcionistické modely

Konekcionistické modely, a konekcionizmus všeobecne, pracuje s metódami, ktoré boli inšpirované ľudským mozgom. Stavebnými jednotkami konekcionizmu sú neuróny, ktorých sila tkvie vo vzájomnom prepojení a interakcii.

Hybridné modely

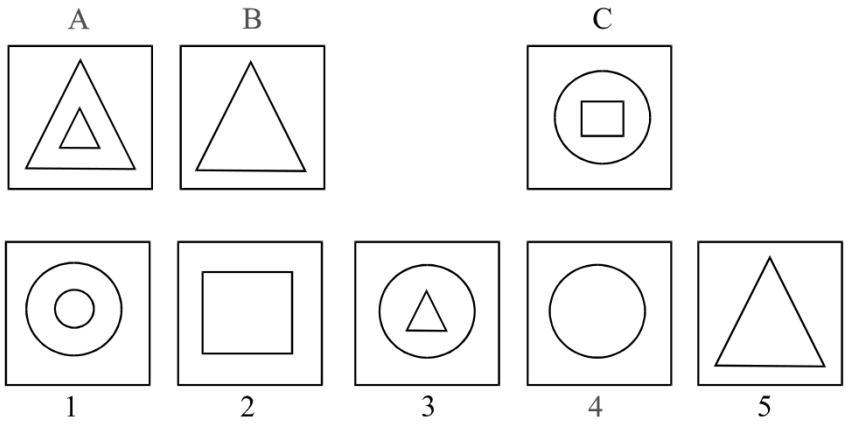
Hybridné modely kombinujú predchádzajúce dva prístupy.

Modely, ktoré nás najviac zaujali, v ďalšej časti detailnejšie popisujeme.

1.2.1 Analogy

Analogy (Evans, 1968) sa radí medzi historicky prvé modely, ktoré dokázali riešiť analógie. Symbolový model riešil geometrické propozičné analógie *A ku B je ako C ku D*, ktoré sa vyskytovali na stredoškolských IQ testoch. Program dostal 3 obrázky na vstupe (*A, B, C*) a 5 možností z ktorých si mal vybrať obrázok *D* tak, že vzťah medzi *A ku B* bude rovnaký ako vzťah *C ku D*. *Analogy* pracoval v dvoch krokoch:

1. Rozpoznanie implicitne reprezentovaných objektov.
2. Hľadanie transformácii, ktoré prerobia *A* na *B* a aplikovanie transformácie na *C*.



Obr. 1.1: Geometrické analógie riešené modelom *Analogy*.

Rozpoznanie implicitne reprezentovaných objektov

Na vstupe programu boli súradnice jednotlivých zložiek objektov, z ktorých program vytvoril explicitnú reprezentáciu objektov a následne detekoval vzťahy medzi jednotlivými objektami (vnorenie, umiestnenie vedľa seba a pod.). Napríklad, ak by sme chceli reprezentovať štvorec a v ňom vnorený kruh, reprezentácia by vyzerala nejako takto:

```
(Object
  (Line 0.5 0.5 1.0 0.5)
  (Line 1.0 0.5 1.0 1.0)
  (Line 1.0 1.0 0.5 1.0)
  (Line 0.5 1.0 0.5 0.5))
(Object
  (Circle 0.75 0.75 0.1))
```

Parametre pre `Line` sú súradnice začiatočného a koncového bodu v rovine a parametre pre `Circle` sú súradnica stredu a polomer. Tento vstup neobsahuje informáciu o tom, že prvý objekt je štvorec a ani to, že kruh je vnorený vo štvorci. Program to všetko musel sám rozpoznať predtým, než začal hľadať analógiu.

Hľadanie transformácií

Po tom, čo bola vytvorená explicitná reprezentácia a vzťahy boli rozpoznané, program hľadal zoznam transformácii, ktoré by mohli prerobiť obrázok A na obrázok B. Nájdené transformácie aplikoval na obrázok C a výsledok hľadal medzi piatimi možnosťami, ktoré sú súčasťou vstupu. Transformácie, ktoré bral do úvahy boli rotácia, škálovanie, translácia, reflexia, odstránenie objektu a ich kombinácie.

Príklad analógie, ktorú riešil *Analogy*, je možné vidieť na obrázku 1.1. Zaujímavé na tomto príklade je, že oficiálna správna odpoveď bola možnosť 4, pretože obrázok B vznikol tak, že sa odstránil vnútorný objekt, no program povedal, že správna odpoveď je ešte aj možnosť 2, lebo obrázok B môže vzniknúť aj tak, že sa odstráni vonkajší objekt a vnútorný sa roztiahne.

1.2.2 SME

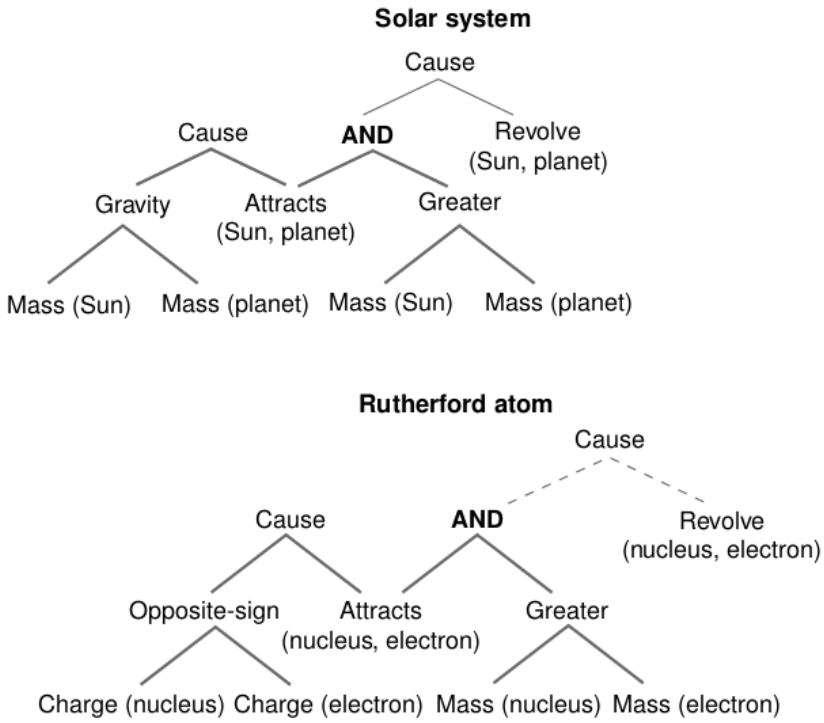
Model *SME* (Falkenhainer a spol., 1986) je symbolovým deterministickým modelom, ktorý nemá mechanizmus učenia sa a ani nepracuje s kontextom. Model je implementáciou *teórií mapovania štruktúr* (Gentner a Bolt, 1980), (Gentner, 1982), (Gentner, 1983), kde Gentnerová opisuje množinu implicitných pravidiel, ktoré ľudia používajú na interpretáciu analógií a podobností. Teórie ponúkajú návod ako namapovať *cieľ* na *zdroj*, potom ako už bol vytiahnutý z pamäte. Nerieši ako *zdroj* identifikovať.

Výpočet v modeli *SME* začína tak, že užívateľ modelu manuálne vytvorí reprezentácie *cieľa* a *zdroja*, pri ktorých použije relevantné informácie o problémoch a nerelevantné informácie vynechá. Potom ako *SME* dostane reprezentácie, program začne skúšať mapovať *zdroj* na *cieľ* tak, že najprv namapuje entity, ktoré majú rovnaký názov a potom prehľadávaním skúša namapovať ostatné entity, tak aby sa zachovali vzťahy medzi objektami.

Program bol úspešne použitý na viacerých problémoch (jeden je ukázaný na obrázku 1.2), no bol zároveň kritizovaný (Chalmers a spol., 1992), že používateľ musí manuálne vytvárať reprezentácie, a tým pádom vkladá priveľa svojej kreativity do programu. To, že program dokáže nájsť analógiu je umožnené prevažne správne namodelovaným vstupom a nie inteligenciou programu.

1.2.3 ACME

Model *ACME* (Holyoak a Thagard, 1989) je prvým konekcionistickým modelom, ktorý podobne ako model *SME* mapuje štruktúru prvého vstupu na štruktúru druhého vstupu. Štruktúry, ktoré sú ručne vytvorené, sú zadané do programu spolu s údajmi o sémantike, kontexte a cieli. Program vytvorí všetky možné dvojice objektov, kde prvá položka dvojice je objekt z prvého vstupu a druhá položka je objekt z druhého vstupu a všetky možné dvojice relácií. Každá dvojica je reprezentovaná jedným neurónom a neuróny sú medzi sebou pospájané spojeniami charakterizovanými váhami. Ak je váha kladná, znamená to, že možnosť existencie vzťahu medzi entitami je vysoká, ak je záporná, možnosť existencie



Obr. 1.2: Stromová reprezentácia solárneho systému a modelu atómu podľa Rutherforda, medzi ktorými program *SME* našiel analógiu. Reprezentácia solárneho systému obsahuje informácie o tom, že Slnko má väčšiu hmotnosť ako planéta a že gravitačne priťahuje planétu, čo zapríčiňuje, že planéta obieha okolo slnka. Reprezentácia modelu atómu obsahuje informácie, že jadro atómu má väčšiu hmotnosť ako elektrón a že jadro priťahuje elektrón. Model *SME* teda usúdil, že existuje analógia medzi priťahovaním planéty s menšou hmotnosťou Slnkom s väčšou hmotnosťou a priťahovaním elektróna s menšou hmotnosťou atómovým jadrom s väčšou hmotnosťou, preto mohol preniesť informácie z modelu solárneho systému do modelu atómu, ktoré sa tam nenachádzajú – elektrón obieha okolo atómového jadra (obrázok prevzatý z French (2002)).

vzťahu je nízka. Nakoniec pomocou relaxačného algoritmu program nájde najvierohodnejšie zobrazenie medzi štruktúrami.

ACME bol úspešne použitý na viacerých problémoch ako napríklad hľadanie analógií medzi príbehmi, medzi problémami, medzi argumentami, medzi politickými situáciami. Za hlavné nedostatky problému sa považuje to, že model nie je psychologický plauzibilný a takisto to, že štruktúry vstupov treba ručne vytvoriť, čím sa vkladá príliš vlastnej kreativity do programu.

1.2.4 LISA

LISA (Hummel a Holyoak, 2005) je neurálne plauzibilný model, ktorý zachytáva flexibilitu a štruktúrovanosť ľudských mentálnych reprezentácii. Model reprezentuje súčasne relácie a vzťahy pomocou predikátových štruktúr, ktoré sú distribuované cez sémantické časti pomocou synchronizovaného pálenia neurónov. Predikátové štruktúry slúžia ako základ pre identifikáciu *zdroja*, analogické mapovanie a analogické usudzovanie. Model zároveň zachytáva obmedzenie ľudskej mysle a dokáže simulovať aj zmeny v myslení spôsobené rôznymi poškodeniami mozgu.

Architektúra modelu je *symbolovo konekcionistická*, čo znamená, že jednotlivé štruktúry sú uložené v rámci neurónovej siete. Vzor aktivácií sémantických neurónov identifikuje, aký objekt tieto neuróny reprezentujú. Napríklad pre objekt *John* by mohli byť aktívne sémantické neuróny *človek*, *dospelý*, *muž*. Objekty sú dynamicky naviazané na ich roly vďaka synchronizovánemu páleniu ich distribuovaných reprezentácií. Rozdielne objekty s priliehajúcimi rolami majú rozdielné synchronizačné rytmy. Výsledné reprezentácie (napr. *John ľúbi Sally*) sú uložené v dlhodobej pamäti. Takáto architektúra využíva výhody klasických symbolových modelov, ktoré sa vyznačujú vysokou štruktúrovanosťou a výhody konekcionistických modelov, ktoré sú zase známe svojou flexibilnosťou.

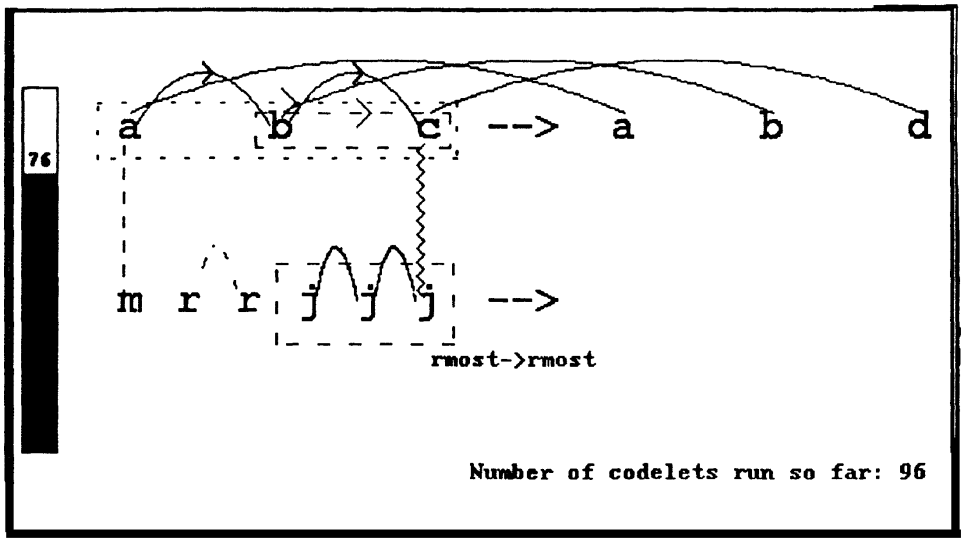
1.2.5 Copycat

Model *Copycat* (Mitchell, 2001) je hybridným, kognitívne plauzibilným modelom, ktorý používa elementy multi-agentového programovania a stochastickosti. Model pracuje na reťazcových analógiách tvaru *A ku B je ako C ku D*. Napríklad platný vstup programu je analógia *abc:abd :: ijk:?*, kde správne riešenie, ktoré má nájsť program, je *ijl*, pretože *cieľový* reťazec vznikne tak, že posledný znak v *zdrojovom* reťazci sa nahradí jeho alfabetickým následníkom. Pretože *Copycat* nepracuje deterministicky, pre rovnaké vstupy môže dávať rôzne riešenia. Riešenia pre vyššie uvedenú analógiu, ktoré mohol program vrátiť, sú:

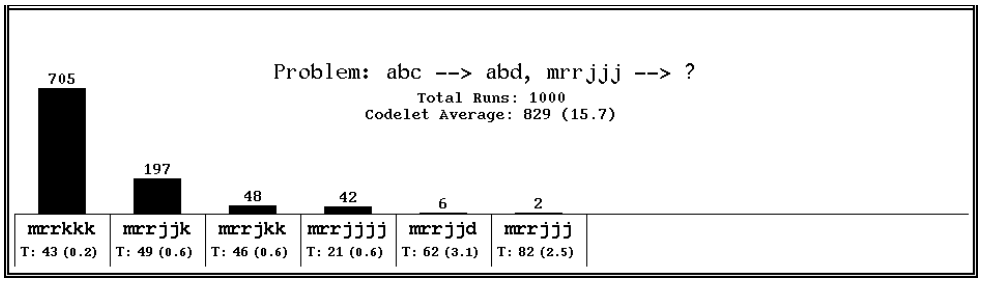
- *ijd* (nahradenie posledného znaku znakom *d*)
- *ijk* (nahradenie všetkých znakov *c* znakom *d*)
- *abd* (nahradenie akýkoľvek reťazec reťazcom *abd*)

Model si vytvára a modifikuje reprezentáciu vstupov počas behu, pričom základ reprezentácie tvorí konceptuálna sieť zložená z vrcholov, ktoré reprezentujú koncepty, a hrán, ktoré reprezentujú vzťahy medzi jednotlivými konceptmi. Sieť používa dynamickú

aktivačnú hodnotu vrcholov, ktorá určuje ako veľmi je aktuálny stav relevantný pre analogický problém. Konceptuálnu sieť modifikujú paralelný agenti (nazvaní *codelety*), ktorí volia také modifikácie na základe pravdepodobností, ktoré by mohli priblížiť aktuálny stav siete ku riešeniu. Na začiatku výpočtu majú všetky akcie rovnakú pravdepodobnosť, že budú vybrané, no s časom klesá pravdepodobnosť, že sa vyberie nejaká nová akcia. Pravdepodobnosť určuje *teplota*, ktorá predstavuje usporiadanosť systému, a ktorá postupne klesá. Obrázok 1.3 približuje správanie programu a obrázok 1.4 zobrazuje výsledky rovnakého problému na viacerých simuláciách.



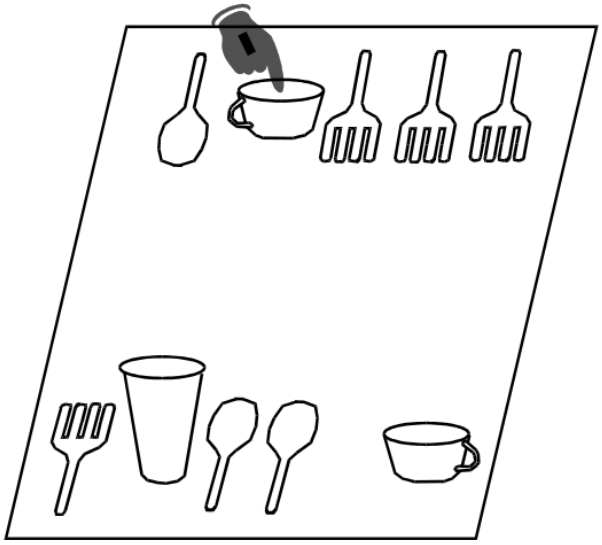
Obr. 1.3: Stav programu *Copycat* pri hľadaní riešenia analógie *abc:abd :: mrrjj:?*. Vľavo od scény je teplomer s hodnotu 76, čo znamená, že je vysoká pravdepodobnosť vykonania modifikácií v scéne. Súvislé čiary predstavujú silné asociácie, ktoré si agenti vytvorili, a prerušované čiary slabé asociácie, ktoré sa neskôr zmenia na silné, alebo sa zrušia. Štvorce predstavujú skupiny (tiež sú prerušované, alebo celé podľa toho, aká pravdepodobná je daná skupina), ktoré si agenti stihli všimnúť a ktoré skúsia nejakým spôsobom využiť v ďalších krokoch výpočtu (obrázok prevzatý z Mitchell (2001)).



Obr. 1.4: Histogram výsledkov programu počas 1000 zbehnutí (obrázok prevzatý z Mitchell (2001)).

1.2.6 Tabletop

Tabletop (French a Hofstadter, 1992) je príbuzný model *Copycatu*, ktorý je rovnako hybridným, stochastickým modelom, ktorý používa multi-agentový prístup na riešenie problému. *Tabletop* bol navrhnutý pre jeden konkrétny problém: určenie príbora, prípadne niečoho podobného, na jednej strane stola, ktorý bol vybratý na opačnej strane (obrázok 1.5). Model pracuje priamo s konceptmi a bol vybudovaný na idei, že percepcia a vytváranie analógií sú nerozdeliteľne previazané.



Obr. 1.5: Problém riešený programom *Tabletop*. Úlohou je určiť objekt na jednej strane stola, ktorý je analogický označenému objektu na druhej strane stola (obrázok prevzatý z (Blank, 1997)).

U človeka sa analógie objavujú ako dôsledok vysoko-úrovňovej percepcie (Chalmers a spol., 1992), na čo sa kladie dôraz aj v *Tabletope*. V modeli sa pod vysoko-úrovňovou percepciou rozumejú následovné paralelné úlohy:

- pomenovanie objektov na základe jednoduchých kategórii
- pomenovanie objektov na vyšších úrovniach abstrakcie, ktoré už boli pomenované na základe jednoduchých kategórii
- hierarchické zgrupovanie na základe pozícií, štruktúr a konceptov
- vytváranie dvojíc, pozostávajúcich z objektu a jeho protikladu na druhej strane, na základe pozícií, štruktúr a konceptov
- priradenie časovo-premenlivých váh objektom a váh dvojiciam
- kompetícia medzi percentuálnymi štruktúrami

Každú úlohu vykonáva samostatný agent, ktorý pracuje na oboch stranách stola. Akcie sú vykonávané s nejakou pravdepodobnosťou, ktorá záleží od toho, či je aktuálny objekt percepčne zaujímavý, alebo nie. To či je objekt zaujímavý určujú faktory ako relatívna pozícia od označeného objektu, štrukturálna, či konceptuálna vzdialenosť označeného objektu, skupina označeného objektu, veľkosť skupiny, pozícia skupiny, či skupina má svoj komplement a pod. Tieto parametre sa časom menia, takže menia sa aj váhy objektov a dvojíc. Nakoniec je objekt analogický označenému ten, ktorý má najvyššiu váhu. *Tabletop* sa považuje za psychologicko-plauzibilný model.

Kapitola 2

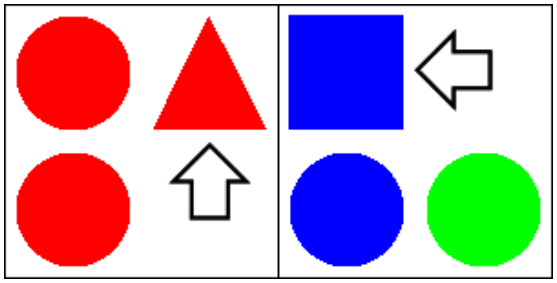
Geometrické analógie

V tejto kapitole špecifikujeme problém, ktorým sa budeme zaoberať. Vybrali sme si problém geometrických analógií, ktorý použil aj Blank (1997) vo svojej dizertačnej práci a ktorý sa vyskytuje často aj na IQ testoch. Dôvod prečo sme si tento problém vybrali je ten, že radi by sme porovnali dosiahnuté výsledky s Blankom a navyše mnohým, vrátane nás, môže pripadať problém zaujímavý a nie úplne triviálny.

2.1 Zadefinovanie problému

Uvažujme takýto problém. Máme dve scény pozostávajúce z geometrických objektov rôznych tvarov a farieb. Prvá scéna má označený jeden objekt, ktorý sa nejakým spôsobom líši od ostatných objektov. Úlohou je nájsť túto unikátnu črtu označeného objektu a označiť objekt v druhej scéne, ktorý sa bude líšiť od svojich ostatných objektov presne v tej iste črte. Ide o hľadanie analógií medzi geometrickými obrazcami.

Ukážka problému je na obrázku 2.1, kde ľavá časť obrázku je spomínaná prvá scéna a pravá časť obrázku druhá scéna. Šípka v prvej scéne určuje objekt, ktorý je predmetom



Obr. 2.1: Príklad geometrickej analógie.

nášho záujmu. Šípka v druhej scéne ukazuje správnu odpoveď (analogický objekt), ktorá nie je k dispozícii. V experimentoch sme sa obmedzili na tri rôzne farby (červená, modrá, zelená) a na tri rôzne tvary objektov (kruh, štvorec, trojuholník). Každá scéna obsahuje práve tri objekty a každý objekt sa nachádza na jednej zo štyroch možných pozícií (2×2).

2.1.1 Typy geometrických analógií

Na obrázku 2.1 je ku červenému trojuholníku vľavo analogický modrý štvorec vpravo, pretože oba sa líšia od ostatných dvoch tvarom. Iná situácia by bola, ak by sa objekty líšili farbou, alebo ak by výsledná scéna vznikla zrkadlovým prevrátením pôvodnej scény, pričom objekty by sa nezmenili. Z toho jasne vyplýva, že existuje viacero *typov*, alebo *kategórií* geometrických analógií a je na nás, ktorým typom sa budeme venovať. My sme si vybrali tie isté typy, ktoré použil Blank vo svojej dizertačnej práci a pridali sme ďalší typ (**symetrické analógie**), ktorý Blank nepoužil[1]. Ide o tieto typy:

- **rotačné analógie** (výsledná scéna vznikla rotáciou pôvodnej scény)
- **symetrické analógie** (výsledná scéna vznikla prevrátením pôvodnej scény)
- **kategorické analógie** (analogický objekt sa líši farbou, alebo tvarom od ostatných)

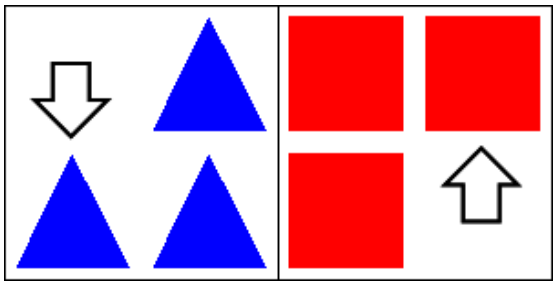
Rotačné analógie

V tomto type analógie je celá scéna otočená o 90°, 180°, 270°, alebo žiadna rotácia nie je aplikovaná (vtedy sú obe scény ekvivalentné). Príklad takejto analógie je ukázaný na obrázku 2.2, kde modrý trojuholník vľavo dole je analogický červenému štvorcu vpravo hore, pretože ak by sme ľavú scénu otočili o 180°, získali by sme rovnaké umiestnenie objektov. Farba a tvar objektov sú pri rotačných analógiách ignorované, pretože použitím viacerých farieb, príp. tvarov, v jednej scéne, by mohli vznikať nejednoznačnosti.

Skúsme sa zamyslieť, koľko môže existovať všetkých takýchto rotačných analógií. Keďže objekty na jednej scéne majú rovnakú farbu a rovnaký tvar, počet možností, ktorými možno zvoliť jeden objekt je 9 (3 farby × 3 tvary). Ďalej sú 4 spôsoby ako umiestniť tri

[1] Blank síce použil symetrické analógie, ale použil ich v kombinácií s kategorickými analógiami, pri ktorých kategorické analógie symetricky prevrátil. Domnievame sa, že sa dopustil chyby, pretože súčasne použil aj kombináciu kategorických a rotačných analógií, pri ktorých kategorické analógie otočil. Tým pádom vzniknú nejednoznačnosti, pretože analogický objekt vo výslednej scéne je iný ak sa pôvodná scéna zrotuje a iný ak sa vytvorí zrkadlový odraz pôvodnej scény. Navyše Blank tvrdil, že jeho model sa túto kombináciu nedokázal naučiť, čo nie je prekvapujúce, pretože úloha nemá riešenie a nedokázal by ju vyriešiť ani človek.

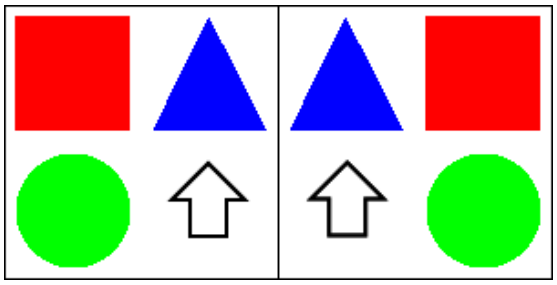
objekty[2] a 3 možnosti označenia jedného objektu. Existuje teda $9 \times 4 \times 3 = 108$ rôznych scén a pre každú scénu existuje $9 \times 4 = 36$ analogických scén (tu už neoznačujeme jeden objekt). Kompletná množina týchto analogických rotačných dvojíc, ktorá bola vygenerovaná pri trénovaní a testovaní, má veľkosť $108 \times 36 = 3888$.[3]



Obr. 2.2: Príklad rotačnej analógie.

Symetrické analógie

Symetrické analógie sú veľmi podobné rotačným analógiám s tým rozdielom, že výsledná scéna nevznikne rotáciou pôvodnej scény, ale prevrátením scény horizontálne, vertikálne, alebo vzhľadom na diagonálu, ktorá prechádza ľavým dolným a pravým horným bodom. Príklad je na obrázku 2.3, kde výsledná scéna vznikla prevrátením vertikálne okolo osi Y.



Obr. 2.3: Príklad symetrickej analógie.

Pri symetrických analógiách sme sa neobmedzili na jednu farbu a jeden typ objektu, ako pri rotačných analógiách, tým pádom veľkosť kompletnej množiny symetrických analógií

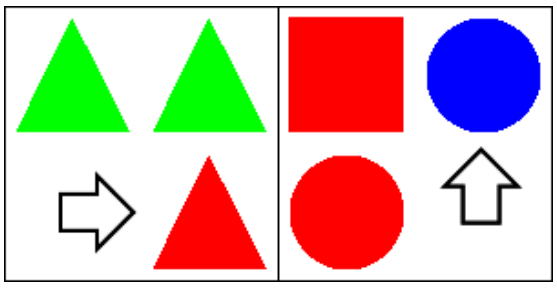
[2] Ide o počet trojprvkových podmnožín (3 objekty) nad štvorprvkovou množinou (4 pozície kam umiestniť objekty), čo je $\binom{4}{3} = 4$.

[3] Blank v skutočnosti vygeneroval 46656 rotačných dvojíc, pretože on sa neobmedzil na použitie iba jednej farby a jedného tvaru v scéne. My sme to urobili preto, lebo radi by sme skúsili naučiť sieť viac typov analógii. Ak by sme mali v rotačnej analógií viacero farieb a tvarov, mohli by vznikať nejednoznačnosti pri trénovaní siete súčasne s iným typom analógií.

je podstatne väčšia. Počet možností ako zvoliť jeden objekt je 9 (3 farby × 3 tvary) a počet možností zvolenia 3 objektov je $9^3 = 729$. Umiestniť objekty na scénu sa dá 4 spôsobmi a označiť objekt sa dá 3 spôsobmi. Výsledná množina všetkých možných scén má potom veľkosť $729 \times 4 \times 3 = 8748$. Keďže sme si zvolili 3 rôzne prevrátenia, výsledná množina všetkých dvojíc má veľkosť $8748 \times 3 = 26244$.

Kategorické analógie

Pri kategorických analógiách platí to, že analogický objekt je práve ten, ktorý sa líši od ostatných dvoch svojou farbou, alebo tvarom. V pôvodnej scéne existuje len práve táto jedna črta, ktorá rozlišuje objekt, pričom v cieľovej scéne sú už dve rôzne črty. Okrem inej farby (resp. tvaru) je použitý aj iný tvar (resp. farba) niektorého iného objektu. Dôvod prečo je to tak je ten, že chceme, aby si sieť v pôvodnej scéne všimla, že podstatná črta je farba (resp. tvar) a aby potom dokázala túto vedomosť aplikovať na cieľovú scénu. Sieť by si mala naučiť, že zaujímavá je práve jedna črta a tú druhú treba ignorovať. Príklad analógie, kde je analogický objekt ten, ktorý sa líši farbou od ostatných dvoch, je na obrázku 2.4, alebo na obrázku 2.1, kde sa naopak líši tvarom.



Obr. 2.4: Príklad farebnej analógie.

Pri generovaní zdrojovej scény je dôležité zmeniť jednu črtu a ponechať tú druhú. Počet možných výberov jednej črty, na ktorú sa zameriavame, použitých v jednej scéne[4] je teda 6 a počet výberov druhej črty je 3. Jeden objekt označíme troma spôsobmi a objekty vieme umiestniť štyrmi spôsobmi. Spolu to vychádza $6 \times 3 \times 3 \times 4 = 216$ scén. Cieľové scény sa generujú rovnako s tým rozdielom, že na celej scéne sú 2 možnosti druhej črty, pričom označený objekt má druhú črtu spoločnú s aspoň jedným objektom. Tu nastávajú prípady, že buď všetky objekty majú rovnakú druhú črtu, alebo prvý neoznačený objekt má inú druhú črtu (2 možnosti ako zmeniť druhú črtu), alebo druhý neoznačený objekt má inú

[4]Tri možnosti ako vybrať farbu (resp. tvar) pre označený objekt a dve možnosti výberu farby (resp. tvaru) ostatných objektov.

druhú črtu. Spolu máme 5 možností ako modifikovať scénu. Cieľová množina má potom veľkosť 216 $\times 5 = 1080$ a výsledná množina analogických dvojíc má veľkosť $216 \times 1080 \times 2 = 466560$ (2 preto, lebo si vyberáme, že analógia sa týka farby, alebo tvaru).

2.2 Reprezentácia scény

Po tom, čo bol problém geometrických analógií zadefinovaný, nastáva otázka ako reprezentovať scény s geometrickými objektami. V zásade existujú dva spôsoby, ktorými môžeme reprezentovať dáta: **implicitne** a **explicitne**, ktoré sú detailnejšie vysvetlené v ďalšej časti.

2.2.1 Implicitná reprezentácia

V implicitnej reprezentácii nie sú štruktúry objektov a vzťahy medzi objektami exaktne opísané. Reprezentácia špecifikuje črty, ale nehovorí aké sú vzťahy medzi črtami a ani ktoré črty patria objektu. Napríklad, ak geometrický objekt nie je definovaný množinou bodov, ale je daný farbami v bitmape. Motivácia za implicitnou reprezentáciou je tá, že v reálnom svete sa explicitné reprezentácie príliš nevyskytujú a človek väčšinou pracuje v doméne implicitných reprezentácií. Ak si človek vytvára analógie, na vstupe má implicitnú reprezentáciu danej situácie, z ktorej si vyextrahuje potrebné črty a tie si potom vie zobraziť medzi sebou tak, aby sa vytvorila analógia. Ak by mal explicitnú reprezentáciu, znamenalo by to, že črty už nie je potrebné extrahovať, čomu sa problém vytvárania analógií podstatne zjednoduší.

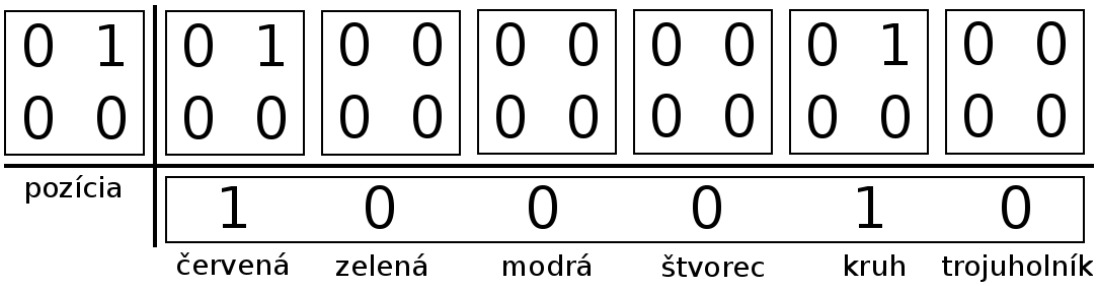
Mitchell a Hofstadter (1995) uvádzajú, že percepcia situácie je dôležitým komponentom pri vytváraní analógií a mala by byť zahrnutá aj vo výpočtovom modeli, ktorý rieši analógie. Uvedomenie si, že percepcia je súčasťou vytvárania analógií bol jeden z najpodstatnejších prínosov kognitívnej vedy pri chápaní analogického uvažovania u ľudí.

Tenzorová reprezentácia

Blank sa pri vytváraní tenzorovej reprezentácie inšpiroval prácou Halford a spol. (1994), kde autori používali tenzory na reprezentáciu predikátov a argumentov. Napríklad pre vetu: „Michal má sestru Annu" by vytvorili dva vektory reprezentujúce objekty *Michal* a *Anna* a jeden vektor reprezentujúci reláciu *má sestru*. Takýto vektor by pozostával prevažne z núl a na jednej pozícií by bola jednotka, ktorá by reprezentovala daný objekt, alebo vzťah. Na oba vektory reprezentujúce objekty by bol použitý **vonkajší súčin** (outer

product)[5], čím by vznikol tenzor druhého rádu, na ktorý by bol znova aplikovaný vonkajší súčin s vektorom, ktorý predstavuje reláciu. Výsledok by bol tenzor tretieho rádu, ktorý implicitne reprezentuje vyššie uvedenú vetu.

Blank použil tento tenzorový prístup, ktorým zakódoval pozíciu a črty geometrických objektov v jednej scéne. Jeho prínos v tenzorovej reprezentácií bol v oddelení pozície objektu od čŕt objektu. Takto dostal maticu 2×2, ktorá reprezentovala pozíciu jedného objektu a vektor dĺžky 6, ktorý hovoril, ktoré črty sú aktívne. Vonkajším súčinom matice a vektora získal trojrozmernú maticu (tenzor tretieho rádu), na ktorý sa môžeme dívať ako na 6 matíc. Príklad zakódovania červeného kruhu v pravom hornom rohu je na obrázku 2.5.



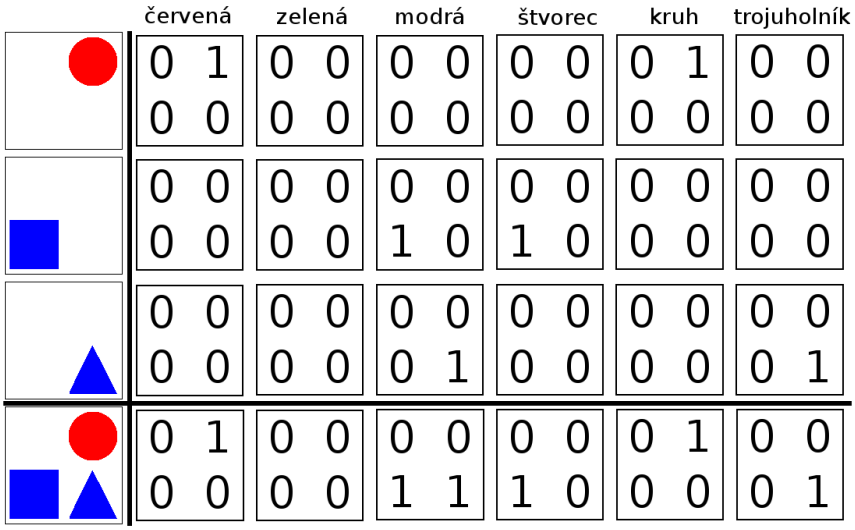
Obr. 2.5: Tenzorová reprezentácia červeného kruhu.

Táto tenzorová reprezentácia je vytvorená pre každý objekt na scéne a matice, ktoré reprezentujú tú istú črtu, sú sčítané. Detailná ukážka je na obrázku 2.6, ktorý demonštruje zakódovanie scény pozostávajúcej z červeného kruhu, modrého štvorca a modrého trojuholníka.

Týmto spôsobom sa zakóduje celá scéna, ale pre geometrické analógie je potrebné ešte reprezentovať označený objekt. Pretože tenzorová reprezentácia špecifikuje všetky objekty, vrátane označeného, nie je nutné špecifikovať ešte raz označený objekt, ale stačí iba jeho pozícia. Pozícia objektu sa zaznamenáva v matici 2×2, ako už bolo ukázané na obr. 2.5. Celá vstupná vrstva teda pozostáva zo 6 matíc reprezentujúcich čŕt objektov a z 1 matice určujúcej pozíciu označeného objektu.

Je známe, že oddelenie pozície objektu od jeho čŕt pri spracovaní percepcie je biologicky plauzibilné. Napríklad Rao a spol. (1997) monitorovali prefrontálnu mozgovú kôru u makakov a objavili neuróny, ktoré boli aktívne pri samotnom objekte (tzv. *what* neuróny) a iné neuróny, ktoré boli aktívne, ak sa objekt posunul (tzv. *where* neuróny). Iné zauj-

[5]Vonkajší súčin dvoch (stĺpcových) vektorov (ktoré môžu mať rôzne dĺžky) je definovaný ako $\mathbf{x} \otimes \mathbf{y}^T$, a jeho výsledkom je matica.

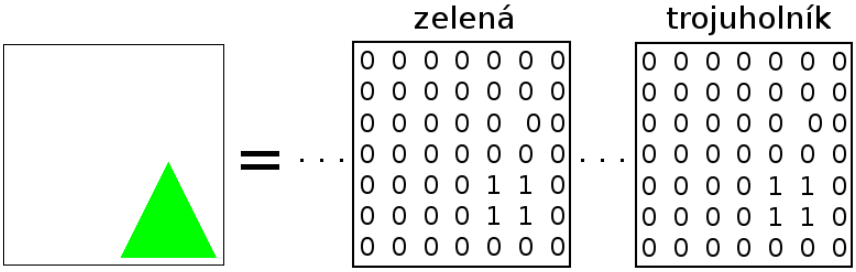


Obr. 2.6: Tenzorová reprezentácia celej scény.

mavé štúdie, ktoré potvrdzujú myšlienku oddelenosti pozície objektu od jeho čŕt sú napr. (Landau a Jackendoff, 1993), (Kosslyn a spol., 1989), (Ungerleider a Mishkin, 1982).

Zväčšená tenzorová reprezentácia

Neskôr v kapitole 3 uvádzame, že Blankov model musí mať skrytú vrstvu rovnako veľkú ako matica na vstupnej vrstve, ktorá kóduje pozíciu označeného objektu. Ak by sme použili tenzorovú reprezentáciu, ktorá používa matice veľkosti 2×2, znamenalo by to, že skrytá vrstva by obsahovala 2×2=4 skryté neuróny. Simulácie ukázali, že 4 neuróny zďaleka nevystačujú a je potrebné nejakým spôsobom zväčšiť veľkosť matíc, aby sa mohla zväčšiť aj skrytá vrstva.



Obr. 2.7: Zväčšená tenzorová reprezentácia zeleného trojuholníka.

Blank to vyriešil tak, že maticu 2×2 nahradil maticou 7×7, kde namiesto jednej jednotky použil štyri, ktoré určujú aktívne črty. Okraj matice, stredný riadok a stredný stĺpec je konštantný a obsahuje vždy samé nuly. Ostatné miesta v matici majú taký istý účel ako

miesta v matici 2×2. Príklady týchto matíc sú na obrázku 2.7, kde je použitá rozšírená tenzorová reprezentácia zeleného trojuholníka umiestneného na scéne vpravo dole. Matice, ktoré nie sú podstatné (obsahujú iba samé nuly) sú vynechané.

2.2.2 Explicitná reprezentácia

Pre explicitnú reprezentáciu dát je charakteristická vysoká organizovanosť a exaktný spôsob uloženia informácií. Väčšina údajov je v počítači takto ukladaná. Symbolové modely vytvárania analógií používali tiež tento typ reprezentácie, za čo boli neskôr kritizované (Chalmers a spol., 1992). Prvý bod kritiky spočíval v používaní explicitnej reprezentácie ako sme už uviedli v predchádzajúcej časti. Ďalším problémom bolo to, že ak si program vytváral explicitnú reprezentáciu, môže existovať viacero ekvivalentných spôsobov ako reprezentovať situáciu, pričom nie všetky reprezentácie umožnia v ďalšom kroku výpočtu vytvoriť analógiu. Príkladom môže byť geometrická analógia z obrázka 2.1, ktorú možno explicitne reprezentovať[6] ako:

```
zdrojová-scéna:
  unikátny-objekt(červený trojuholník)
  ostatné-objekty(červený kruh, červený kruh)
cieľová-scéna:
  unikátny-objekt(modrý štvorec)
  ostatné-objekty(modrý kruh, zelený kruh)
```

Vtedy výpočtový model, ktorý rieši analógie, by hravo vedel zobraziť unikátny objekt zo zdrojovej scény na unikátny objekt z cieľovej scény a ostatné objekty na ostatné objekty, vďaka čomu by bola analógia úspešne vytvorená. Lenže na scéne neexistuje objekt, ktorý sa líši od ostatných dvoch iba v jednej črte a preto je možné vytvoriť viacero reprezentácií. Problém nastáva vtedy, ak by bola použitá táto reprezentácia (ktorá je ekvivalentná s tou predchádzajúcou):

```
zdrojová-scéna:
  unikátny-objekt(červený trojuholník)
  ostatné-objekty(červený kruh, červený kruh)
cieľová-scéna:
  unikátny-objekt(zelený kruh)
  ostatné-objekty(modrý kruh, modrý štvorec)
```

[6]Pozície objektov v reprezentácií nie sú uvedené, nakoľko nie sú dôležité v tomto príklade.

Je vidieť, že pri použití tejto reprezentácií by správna analógia nebola vytvorená.

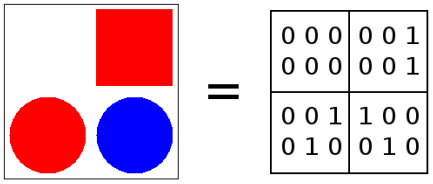
Naša explicitná reprezentácia

Naša explicitná reprezentácia začína nad úrovňou problému viazania (angl. binding problem)[7], čo znamená, že každá črta je už naviazaná na prislúchajúci objekt. Dôvod prečo používame explicitnú reprezentáciu napriek kritike je ten, že pri našej reprezentácii neexistujú nejednoznačnosti ako pri predchádzajúcej reprezentácii. Scénu je možné reprezentovať len jediným možným spôsobom. Navyše ak by používanie explicitnej reprezentácie zvýšilo výkon siete, potom má zmysel používať explicitnú reprezentáciu namiesto implicitnej. V takomto prípade by bolo nutné pretransformovať implicitnú reprezentáciu, ktorá je považovaná za jedinú korektnú reprezentáciu pri analógiách, na našu explicitnú reprezentáciu.

Naša reprezentácia používa 6 bitov na opis čŕt objektu. Prvé tri bity určujú farbu a ďalšie tri bity tvar objektu. Prvý bit reprezentuje modrú farbu, druhý bit zelenú farbu, tretí bit červenú farbu. Z ďalších troch bitov prvý bit predstavuje trojuholník, druhý bit kruh a tretí bit štvorec. Obrázok 2.8 znázorňuje toto kódovanie.

Obr. 2.8: Explicitná reprezentácia zeleného štvorca.

Reprezentácia sa vytvorí pre každý roh v scéne a výsledné 4 vektory charakterizujú celú scénu. Ak sa v danom rohu nič nenachádza, použije sa nulový vektor. Príklad celej scény reprezentovanou týmto spôsobom je na obrázku 2.9 (vektor je kvôli prehľadnosti rozdelený na dva riadky).



Obr. 2.9: Explicitná reprezentácia celej scény.

Pozíciu označeného objektu špecifikujeme rovnako ako pri tenzorovej reprezentácií.

[7]Problém viazania je otázka ako určiť, ktoré črty patria jednotlivým objektom pri percepcii. Napríklad pri jedení človek vníma rôzne črty jedla, ako sú chuť, vôňa, farba a pozícia jedla. Vo chvíli keď tieto črty narazia na senzorické receptory, sú spracované rôznymi časťami mozgu a zároveň sú reprezentované rozdielne. Otázka teda znie, ako mozog určuje, ktoré črty patria tomu istému objektu. Problém sa nazýva problém viazania. (Holcombe, 2009)

Kapitola 3

Model Analogator

Konekcionistický model *Analogator* bol navrhovaný tak, aby sa dokázal učiť sa analógie typu *časť–celok* (typ analógie vysvetlený v sekcii 1.1) na problémoch z rôznych domén. Zásadný rozdiel medzi *Analogatorom* a inými modelmi je ten, že *Analogator* nemá architektúru postavenú na prácu s analógiami. Je to model, ktorý sa postupne naučí analogicky uvažovať. Ďalší rozdiel je ten, že model nie je doménovo–závislý (ako napr. model od Jani a Levine (2000), alebo model *Tabletop* od French a Hofstadter (1992)). *Analogator* sa dokáže naučiť analogický problém z akejkoľvek domény, ak analógia je typu *časť–celok*.

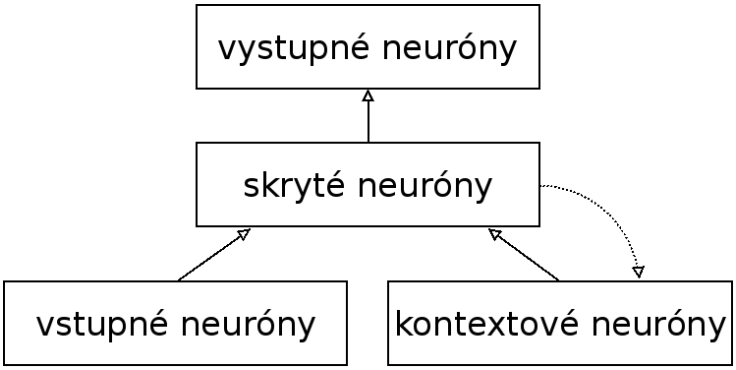
3.1 Model AB1-X

Pod označením AB1-X (Analogator Basic 1 Hidden) budeme rozumieť základnú, nami nemodifikovanú, verziu modelu *Analogator*, ktorá opisujeme v následujúcich častiach. Pretože reprezentácia vstupu nesúvisí s architektúrou modelu a pre problém geometrických analógií máme dve možné reprezentácie (implicitná/explicitná), symbol X značí ľubovoľnú reprezentáciu z týchto dvoch. V kapitole 4, kde sa opisujú experimenty, je namiesto X použitý symbol I, ktorý predstavuje implicitnú reprezentáciu, alebo symbol E predstavujúci explicitnú reprezentáciu. Skratky sme zaviedli z dôvodu zvýšenia prehľadnosti medzi jednotlivými modelmi.

3.1.1 Jednoduchá rekurentná sieť

Blank (1997) sa vo svojej dizertačnej práci venoval téme ako naučiť program, aby dokázal vidieť analógie. V práci predstavil model *Analogator*, ktorý je postavený na jednoduchej rekurentnej (neurónovej) sieti (angl. simple recurrent network – Elman (1990)). Sieť pou-

žíva 3 vrstvy neurónov: vstupnú, skrytú a výstupnú, pričom súčasťou vstupnej vrstvy sú kontextové neuróny, v ktorých sú uložené aktivácie neurónov skrytej vrstvy z predchádzajúceho časového kroku. Architektúra siete je na obrázku 3.1.



Obr. 3.1: Architektúra jednoduchej rekurentnej siete.

Základná funkčná jednotka neurónovej siete je neurón, ktorý je spojený s inými neurónmi cez synaptické spojenia charakterizovaných váhami. Aktivácia neurónov je daná vzťahom

$$y_i(t+1) = f(\sum_j w_{ij} x_j(t))$$

kde x_j predstavuje vstupný neurón neurónu y_i a w_{ij} predstavuje váhu medzi týmito neurónmi. Ako aktivačná funkcia sa zvykne používať funkcia *sigmoida*

$$f(u) = 1/(1+\exp(-u))$$

ktorá je použitá aj v našej práci. Na rozdiel od dopredných neurónových sietí, pri ktorých sa aktivácia šíri iba jedným smerom (smerom od vstupnej vrstvy ku výstupnej vrstve), pri rekurentných sieťach sa aktivácia šíri aj smerom opačným, konkrétne zo skrytej vrstvy do kontextovej.

Algoritmus 3.1 Algoritmus spätného šírenia chyby

Vstup: trénovací vzor x, požadovaný výstup d
$y :=$ Vyrátaj výstup(x)
Vyrátaj chybu(d, y)
Vyrátaj Δw_h pre všetky váhy medzi výstupnou a skrytou vrstvou
Vyrátaj Δw_i pre všetky váhy medzi skrytou a vstupnou vrstvou
Uprav váhy, aby sa chyba zmenšila

Váhy siete sú v procese trénovania siete nastavované algoritmom spätného šírenia chyby (angl. backpropagation algorithm) (algoritmus 3.1), ktorý pre každý trénovací vzor vyráta

aktiváciu na výstupnej vrstve, určí chybu, ktorú sieť urobila a následne upraví váhy siete tak, aby sa chyba pre trénovací vzor zmenšila. Chyba na jednotlivých neurónoch je vypočítaná vzťahmi:

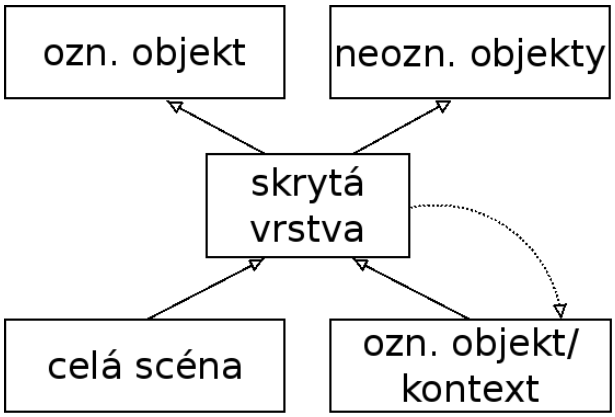
- $\delta_i = (d_i - y_i)(1 - y_i)y_i$ na výstupnej vrstve
- $\delta_k = (\sum_i w_{ik}\delta_i)(1 - h_k)h_k$ na skrytej vrstve

Následne sa váhy modifikujú podľa týchto pravidiel (hodnota α je rýchlosť učenia a μ je moment):

- $\Delta w_{ik}(t+1) = \alpha\delta_i h_k + \mu\Delta w_{ik}(t)$ váhy medzi skrytou a výstupnou vrstvou
- $\Delta v_{kj}(t+1) = \alpha\delta_k x_j + \mu\Delta v_{kj}(t)$ váhy medzi vstupnou a skrytou vrstvou

3.1.2 Architektúra modelu AB1-X

Model AB1-X (základný model) má rovnakú architektúru (obrázok 3.2) ako jednoduchá rekurentná sieť opísaná v predchádzajúcej časti. Vstupná vrstva je rozdelená na dve časti, pričom v prvej časti je uložená reprezentácia celej scény a v druhej časti je uložený označený objekt (v prvom kroku výpočtu), alebo kópia skrytej vrstvy (v druhom kroku výpočtu). Výstupná vrstva obsahuje pozíciu označeného objektu a pozície ostatných, neoznačených objektov v celej scéne.



Obr. 3.2: Architektúra modelu AB1-X.

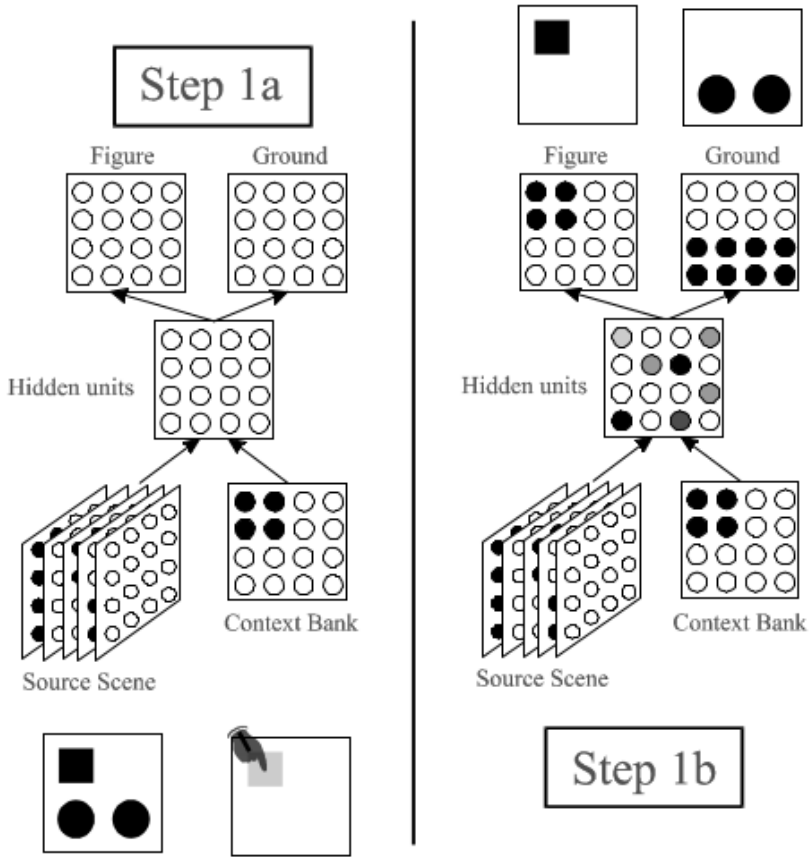
Výpočet pre vytvorenie analógie pozostáva z dvoch krokov:

1. Do prvej časti vstupnej vrstvy je uložená reprezentácia scény (črty objektov a ich pozície) pozostávajúca zo všetkých objektov a do druhej časti (ktorá zároveň zohráva

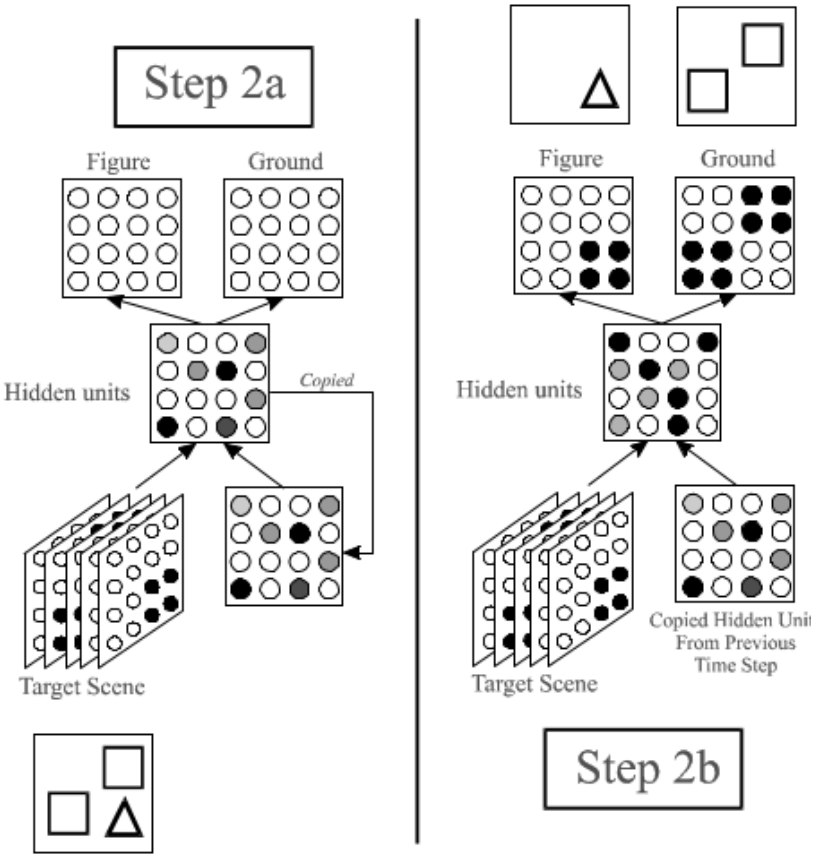
úlohu kontextových neurónov) je uložená pozícia objektu v scéne, ktorý je cieľom záujmu (označený objekt). Následne sa vyrátajú aktivácie skrytej vrstvy a výstupnej vrstvy. Prvá časť výstupnej vrstvy zobrazuje pozíciu označeného objektu a druhá časť zobrazuje pozície zvyšných objektov.

2. Reprezentácia ďalšej scény (medzi ktorou sa hľadá analógia s prvou scénou) je vložená do prvej časti vstupnej vrstvy. Na druhú časť vstupnej vrstvy, ktorá v minulom kroku obsahovala pozíciu označeného objektu, sa uložia aktivácie skrytej vrstvy z predchádzajúceho kroku. Po vyrátaní aktivácií je na výstupnej vrstve pozícia objektu, ktorý je analogický označenému objektu v prvej scéne, a pozície zvyšných objektov.

Príklad vytvorenia analógie je znázornený na obrázkoch 3.3 (prvý krok) a 3.4 (druhý krok).



Obr. 3.3: Prvý krok výpočtu analógie. Na vstupe je scéna pozostávajúca z dvoch kruhov a jedného štvorca, ktorý je zároveň označeným objektom. Sieť má za úlohu rozdeliť scénu na označený objekt a neoznačené objekty (obrázok prevzatý z Blank (1997)).



Obr. 3.4: Druhý krok výpočtu analógie. Na vstupe je scéna pozostávajúca z dvoch štvorcov, jedného trojuholníka a kópie skrytej vrstvy z prvého kroku výpočtu analógie. V tomto kroku má sieť za úlohu rozdeliť scénu rovnakým spôsobom ako bola rozdelená v predchádzajúcom kroku – podľa tvaru (obrázok prevzatý z Blank (1997)).

Obmedzenie, ktoré vyplýva z tejto architektúry je také, že počet neurónov skrytej vrstvy musí byť zhodný s počtom neurónov v kontextovej časti vstupnej vrstvy a tým pádom aj s počtom neurónov pre určenie pozície označeného objektu. Ak by sa zvolil nízky počet neurónov pre označenie objektu, znamenalo by to, že musí byť aj nízky počet neurónov na skrytej vrstve, čo nemusí postačovať pre naučenie siete. Vyšší počet neurónov na skrytej vrstve zase núti pridávať dodatočné a nepotrebné neuróny na vstupnú vrstvu, čím zbytočne stúpajú pamäťové a časové nároky.

3.1.3 Učenie modelu

Pretože výpočet modelu pozostáva z dvoch krokov, učenie modelu v danej iterácií takisto trvá dva kroky. V prvom kroku sa model učí správne rozdeliť scénu na označený

objekt a zvyšné objekty a v druhom kroku sa učí korektne aplikovať túto transformáciu na novú scénu. Oba kroky vyžadujú algoritmus spätného šírenia chyby, ktorý bol opísaný algoritmom 3.1. Pred samotným učením sú všetky váhy nastavené na náhodné hodnoty z intervalu $(-0.05, 0.05)$.

Sieť pri učení vyžaduje dve scény medzi ktorými ide robiť analógiu (nazvané *zdrojová scéna* a *cieľová scéna*), označené objekty v obidvoch scénach a takisto neoznačené objekty. Na vstup sa zadá zdrojová scéna spolu s označeným objektom, vyráta sa prislúchajúci výstup a následne sa váhy upravia, aby bola scéna správne rozdelená na označený objekt a neoznačené objekty. Pripomíname, že na výstupe sa vyskytujú už iba pozície objektov, nie ich črty. V druhom kroku sa nastaví na vstup cieľová scéna a skopírujú sa aktivácie skrytej vrstvy z minulého kroku. Vyráta sa výstup a váhy sa následne pozmenia.

Učenie končí vo chvíli, keď bol dosiahnutý stanovený počet epoch. Výstup siete interpretujeme tak, že všetky výstupné hodnoty zaokrúhlime na 1, alebo 0. Tie potom porovnáme s požadovaným výstupom a ak sú oba výstupy identické, výstup siete považujeme za korektný.

Algoritmus 3.2 Algoritmus učenia modelu AB1-X

Vstup:
zdrojová scéna $s1$, označený objekt $o1$, neoznačené objekty $n1$
cieľová scéna $s2$, označený objekt $o2$, neoznačené objekty $n2$

$vstup := s1 + o1$ ▷ operátor + značí zreťazenie
$vystup := o1 + n1$
Algoritmus spätného šírenia chyby($vstup$, $vystup$)
$kontext :=$ Aktivácie skrytej vrstvy
$vstup := s2 + kontext$
$vystup := o2 + n2$
Algoritmus spätného šírenia chyby($vstup$, $vystup$)

3.2 Model AP1-X

Model AP1-X (Analogator PCA 1 Hidden) sme navrhli s cieľom odstrániť obmedzenie predchádzajúceho modelu, pri ktorom musí byť rovnako veľká skrytá vrstva s časťou vstupnej vrstvy pre určenie pozície označeného objektu. Vyriešené je v tom zmysle, že veľkosť týchto dvoch komponentov prestáva byť na sebe závislá.

3.2.1 Analýza hlavných komponentov

Analýza hlavných komponentov (angl. Principal component analysis, PCA) je algoritmus, ktorý dokáže pomocou lineárnej projekcie redukovať dimenziu dát, ktorých jednotlivé zložky sú potenciálne korelované, na dáta bez korelovaných zložiek tak, že najväčší rozptyl v dátach bude uchovaný. Inými slovami, algoritmus znižuje dimenziu dát – n-rozmerné dáta pretransformuje na dáta k-rozmerné ($n > k$), pričom sa stratí čo najmenej informácie.

Existuje viacero *PCA* algoritmov, ako napr. *Oja's rule* (Oja, 1982), *GHA* (Sanger, 1989), alebo *APEX* (Kung a Diamantaras, 1990). V práci sme si vybrali algoritmus *GHA*, ktorý je popísaný nižšie.

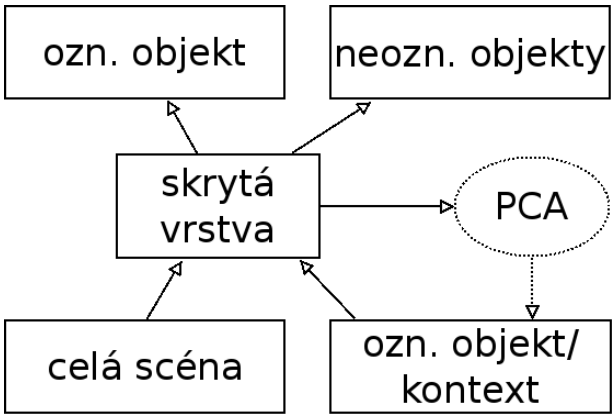
Algoritmus má na vstupe n-rozmerné dáta $\mathbf{x} = (x_1, x_2, ..., x_n)$, ktoré sa postupne učí transformovať na k-rozmerné dáta $\mathbf{y} = (y_1, y_2, ..., y_k)$. Dáta $\mathbf{y}$ sú získané prenásobením dát $\mathbf{x}$ s prislúchajúcimi váhami: $\forall i : y_i = \sum_j w_{ij} x_j$. Učenie algoritmu spočíva v modifikovaní váh matice W. Pred samotným učením sú váhy nastavené na malé náhodné hodnoty. Algoritmus je trénovaný na rôznych vstupoch a pre každý vstup upravuje váhy podľa vzťahu $\Delta w_{ij} := \alpha y_i (x_j - \sum_{k=1}^{i} w_{kj} y_k)$. Parameter α predstavuje rýchlosť učenia. Celý algoritmus je opísaný nižšie (algoritmus 3.3).

Algoritmus 3.3 Algoritmus GHA

Vstup: vzor $\mathbf{x}$, dimenzia vzoru n, nová dimenzia k
for $i := 1$ **to** k **do**
  $y_i := \sum_j w_{ij} x_j$
end for
for $i := k$ **downto** 1 **do**
  for $j := 1$ **to** n **do**
    $\Delta w_{ij} := \alpha y_i (x_j - \sum_{m=1}^{i} w_{mj} y_m))$
  end for
end for
return y

3.2.2 Architektúra modelu AP1-X

Model AP1-X má takmer identickú architektúru s modelom AB1-X. Jediný rozdiel je, že v druhom kroku výpočtu, kedy sa má skopírovať skrytá vrstva do kontextových neurónov, skrytá vrstva prelezie cez *PCA* a výsledok sa uloží do kontextových neurónov. Použitie *PCA* umožňuje meniť veľkosť skrytej vrstvy bez ohľadu na veľkosť vstupnej vrstvy. Nie je dôvod používať na vstupnej vrstve viac neurónov ako je potrebných a k tomu pomôže



Obr. 3.5: Architektúra modelu AP1-X.

práve *PCA*. Stačia 4 neuróny pre časť vstupnej vrstvy, ktorá kóduje pozíciu označeného objektu oproti 49 neurónom, ktoré používal Blank. Model je znázornený na obrázku 3.5.

Výpočet pre model AP1-X vyzerá takto:

1. Vyrátanie výstup siete pre prvú scénu s označeným objektom rovnako ako pri modeli pri AB1-X (sekcia 3.1.2).

2. Transformácia skrytej vrstvy cez *PCA* a uloženie výsledku do časti pre kontextové neuróny. Následne vyrátanie výstupu siete pre ďalšiu scénu.

3.2.3 Učenie modelu

Zavedením *PCA* do modelu bol model obohatený o ďalší učiaci algoritmus. Spolu s algoritmom spätného šírenia chyby má model dva učiace sa algoritmy, ktoré sú nezávisle medzi sebou, ale pritom sa učia na rovnakých dátach. Otázka, ktorá sa prirodzene nastoľuje, je ako algoritmy používať, aby si navzájom neprekážali a aby spolupracovali. Existuje viacero možností ako algoritmy používať. Tú sú niektoré, ktoré sme preskúmali:

1. Paralelné učenie oboch algoritmov počas celej simulácie.

2. Paralelné učenie oboch algoritmov, pričom *PCA* sa učí len **do** nejakej epochy. Po nej sa model učí už iba algoritmom spätného šírenia chyby.

3. Paralelné učenie oboch algoritmov, pričom *PCA* sa učí len **od** nejakej epochy, ale nie od prvej epochy.

4. Učenie samotným algoritmom spätného šírenia chyby. *PCA* sa neučí a je ponechané na vykonávanie nejakej náhodnej lineárnej transformácie.

Simulácie ukázali, že ako najlepšia možnosť sa javí možnosť číslo 1, kde je model trénovaný súčasne oboma algoritmami počas celej simulácie. Ostatné možnosti dávali podstatne horšie výsledky. Dôležité je zároveň správne nastavenie rýchlosti učenia. V simuláciách sa javila najlepšia rýchlosť učenia $\alpha_{PCA} = 0.0001$. Pri vyšších hodnotách sa sieť nedokázala vôbec učiť a dávala vždy 100% trénovaciu chybu. Celý algoritmus je zosumarizovaný v 3.4.

Algoritmus 3.4 Algoritmus pre učenie modelu AP1-X

Vstup:
zdrojová scéna $s1$, označený objekt $o1$, neoznačené objekty $n1$
cieľová scéna $s2$, označený objekt $o2$, neoznačené objekty $n2$
veľkosť skrytej vrstvy n, veľkosť kontextu k

$vstup := s1 + o1$ ▷ operátor + značí zreťazenie
$vystup := o1 + n1$
ALGORITMUS SPÄTNÉHO ŠÍRENIA CHYBY($vstup$, $vystup$)
$kontext :=$ ALGORITMUS GHA($skryta\_vrstva$, n, k)
$vstup := s2 + kontext$
$vystup := o2 + n2$
ALGORITMUS SPÄTNÉHO ŠÍRENIA CHYBY($vstup$, $vystup$)

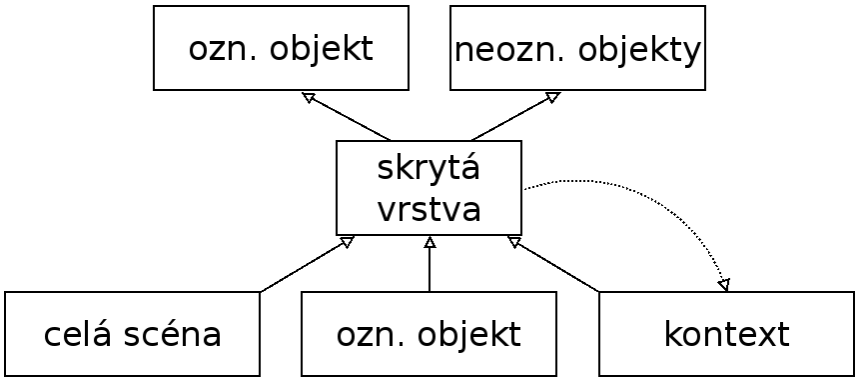
3.3 Model AS1-X

Ďalšia modifikácia základného modelu AB1-X, ktorý predstavujeme v práci, je model pri ktorom je časť vstupnej vrstvy rozdelená na dve časti (Analogator Separated 1 Hidden). Rozdelená je časť, ktorá uchováva informáciu o pozícií označeného objektu a zároveň hrá úlohu kontextových neurónov. To umožňuje takisto riešiť obmedzenie základného modelu, ktoré núti mať skrytú vrstvu a túto časť rovnako veľkú.

3.3.1 Architektúra modelu AS1-X a učenie

Model má rozdelenú časť vstupnej vrstvy zodpovednú za pozíciu označeného objektu a zároveň si pamätá kópiu skrytej vrstvy v druhom kroku výpočtu. Architektúra je zobrazená na obrázku 3.6. Výpočet sa deje nasledujúcim postupom:

1. Vloženie celej scény spolu s pozíciou označeného objektu do vstupnej vrstvy. Kontextové neuróny zostanú nulové. Následné vyrátanie aktivity na výstupnej vrstve.

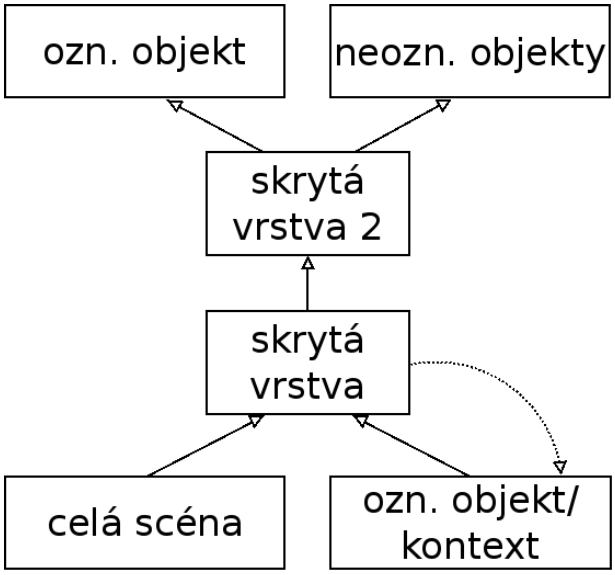


Obr. 3.6: Architektúra modelu AS1-X.

2. Vloženie ďalšej scény spolu so skrytou vrstvou do vstupnej vrstvy. Pozícia označeného objektu sa vynuluje a vypočíta sa aktivita na výstupnej vrstve.

Model AS sa učí rovnako ako základný model (algoritmus 3.2), mení sa iba vstup algoritmu spätného šírenia chyby.

3.4 Model AB2-X



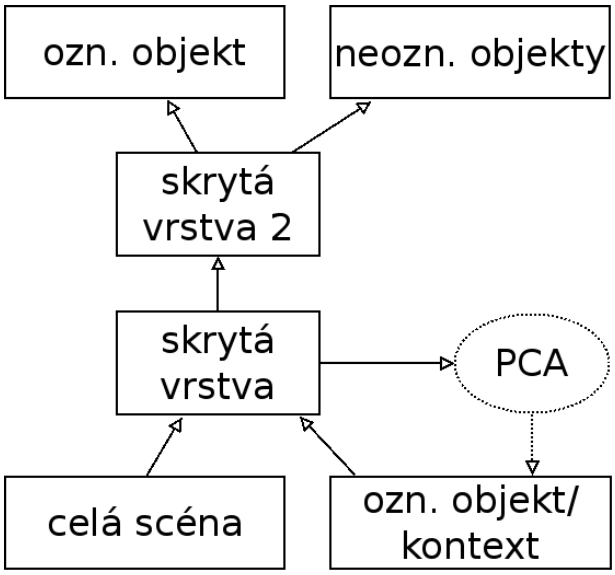
Obr. 3.7: Architektúra modelu AB2-X.

Posledná modifikácia, ktorú sme navrhli v práci, je pridanie novej skrytej vrstvy medzi existujúcu skrytú vrstvu a výstupnú vrstvu (Analogator Basic 2 Hiddens). Toto nerieši spomínané obmedzenie základného modelu, no pomáha to modelu lepšie sa učiť analógie,

rýchlejšie konvergovať a viac generalizovať. Zobrazenie architektúry je možné vidieť na obrázku 3.7. Vyrátanie výstupu a učenie modelu prebieha identicky ako pri základnom modeli (algoritmus 3.2).

3.5 Model AP2-X

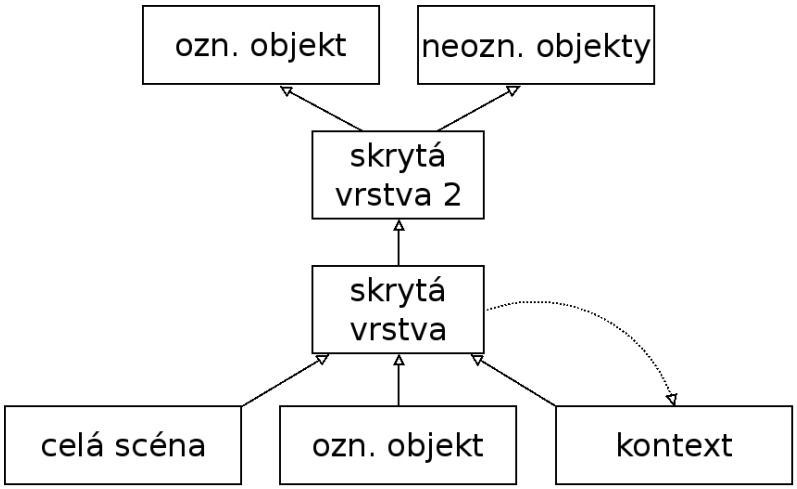
Model vznikol kombináciou modelov AP1-X a AB2-X (Analogator PCA & 2 Hiddens). Ide o model, ktorý pomocou *PCA* transformuje prvú skrytú vrstvu do vstupnej vrstvy a nad ňou má ďalšiu skrytú vrstvu. Architektúra je znázornená na obrázku 3.8.



Obr. 3.8: Architektúra modelu AP2-X.

3.6 Model AS2-X

Posledný predstavený model je model s oddelenou vstupnou vrstvou a novou skrytou vrstvou medzi existujúcou skrytou a výstupnou vrstvou (Analogator Separated & 2 Hiddens), ktorý je možné vidieť na obrázku 3.9.



Obr. 3.9: Architektúra modelu AS2-X.

3.7 Nefunkčné a nepoužité modely

Ak by sme chceli urobiť všetky kombinácie navrhovaných modifikácii modelu *Analogator*, dostali by sme spolu 8 modelov:

$$(PCA, Separated, Hidden)$$
$$(PCA, Separated, \neg Hidden)$$
$$(PCA, \neg Separated, Hidden)$$
$$(PCA, \neg Separated, \neg Hidden)$$
$$(\neg PCA, Separated, Hidden)$$
$$(\neg PCA, Separated, \neg Hidden)$$
$$(\neg PCA, \neg Separated, Hidden)$$
$$(\neg PCA, \neg Separated, \neg Hidden)$$

V predchádzajúcich častiach sme opísali iba 6 modelov a vynechali sme model (PCA, $Separated$, $\neg Hidden$) a model (PCA, $Separated$, $Hidden$). Dôvod vynechania je ten, že nemá zmysel kombinovať modifikáciu PCA s modifikáciou $Separated$. PCA sa použilo preto, aby sa odstránila závislosť medzi veľkosťou skrytej vrstvy a veľkosťou časti vstupnej vrstvy. $Separated$ odstraňuje závislosť takisto, len iným spôsobom. Preto nemá zmysel uvažovať o skombinovaní týchto dvoch modifikácií.

Preskúmali sme ešte model rozšírený o algoritmus samoorganizujúcej sa mapy (angl. self organising map, SOM), ktorý podobne ako PCA redukuje dimenzionalitu dát. Algorit-

mus *SOM* bol použitý na transformáciu skrytej vrstvy do časti vstupnej vrstvy, no takýto model nedokázal fungovať korektne, preto ho v tejto kapitole bližšie neopisujeme.

3.8 Stručný prehľad modelov

Tabuľka 3.1 obsahuje stručný prehľad modelov, ktoré sme v práci navrhli, spolu s reprezentáciou problému geometrických analógií (kapitola 2), aby sme uľahčili orientáciu v ďalšej kapitole venovanej experimentom.

Tabuľka 3.1: Prehľad všetkých navrhnutých modelov

	PCA	*Separated*	*2 × Hidden*	Impl. repr.	Expl. repr.
Model AB1-I				✓	
Model AB1-E					✓
Model AP1-I	✓			✓	
Model AP1-E	✓				✓
Model AS1-I		✓		✓	
Model AS1-E		✓			✓
Model AB2-I			✓	✓	
Model AB2-E			✓		✓
Model AP2-I	✓		✓	✓	
Model AP2-E	✓		✓		✓
Model AS2-I		✓	✓	✓	
Model AS2-E		✓	✓		✓

Kapitola 4

Experimenty

V tejto kapitole predstavujeme naše výsledky, ktoré sme dostali pomocou počítačových simulácií, modelu *Analogator* na probléme geometrických analógií.

4.1 Všeobecné informácie o experimentoch

Pretože naučiť sieť jeden typ analógie je ľahká úloha, testovali sme to iba na základnej verzii *Analogatora*, na modeli AB1-X. Modifikované modely boli použité vždy na dvoch typoch geometrických analógií, pokiaľ v časti venujúcej sa modelu nie je uvedené inak. Tieto dva typy boli **kategorické analógie** a **rotačné analógie**.

Aby sme zachytili približné priemerné správanie daného modelu, model bol trénovaný a testovaný 5 krát a výsledný graf s trénovacou a testovacou krivkou bol priemerom týchto 5 simulácií. Každá jedna simulácia používala rôzne trénovacie a testovacie dáta a trvala rovnaký počet epoch (alebo skončila ak bola dosiahnutá nulová trénovacia chyba). Množina vzorov bola vygenerovaná tak, že sme najprv vygenerovali všetky kategorické analógie (tak ako je to opísané v časti 2.1.1), z ktorých sme náhodne vybrali 20000. Potom sme k nim pridali 3888 rotačných analógií a celú množinu sme premiešali. Z výslednej množiny sme vybrali 90% na trénovanie siete a 10% na testovanie siete.

Pre každý model sme potom určili koľko trénovacích vzorov model potrebuje na to, aby dával minimálnu testovaciu chybu. Postup bol nasledujúci. Množina vzorov mala veľkosť 23888 a bola rozdelená na trénovaciu a testovaciu množinu v rovnakom pomere - 90% trénovacie dáta a 10% testovacie dáta. Testovacia množina zostala po zvolení fixná. Zvolili sme náhodne 10% vzorov trénovacej množiny ($23888 \times 90\% \times 10\% = 2150$ vzorov), na ktorých sa sieť učila, a po naučení sa vyrátala testovacia chyba, ktorá sa uložila. Váhy siete sa

Algoritmus 4.1 Určenie vplyvu veľkosti trénovacej množiny na testovaciu chybu

```
Input: Neurónová sieť ns, Množina vzorov mv
pmv := Premiešaj(mv)
trm := Podmnožina(pmv, 1, 90)              ▷ trénovacia množina veľkosti 90% mv
tem := Podmnožina(pmv, 91, 100)            ▷ testovacia množina veľkosti 10% mv
for v := 10% to 100% step 10% do
    for i := 1 to 5 step 1 do
        Nastav náhodné váhy(ns)
        trmv := Náhodná podmnožina(trm, v)     ▷ náhodná podmnožina veľkosti v
        Trénuj(ns, trmv)
        chyba[i] := Testuj(ns, tem)
    end for
    priemer := Priemer(chyba)
    odchýlka := Smerodajná odchýlka(chyba)
    Zapíš do súboru(priemer, odchýlka)
end for
```

nastavili na náhodné a proces sa opakoval s inými vzormi, tiež o veľkosti 10% všetkých trénovacích dát. Postup bol zopakovaný spolu 5-krát a priemer týchto testovacích chýb, spolu so štandardnou odchýlkou, sa uložil do súboru. Tým sa získalo približné správanie modelu trénovaného 2150 vzormi. Celý algoritmus bol vykonaný pre veľkosti $10\%, 20\%, ..., 100\%$ trénovacej množiny a vznikol tak graf, ktorý znázorňoval vplyvy rôznych veľkostí trénovacej množiny na testovaciu chybu. Celý algoritmus je znázornený vyššie (algoritmus 4.1).

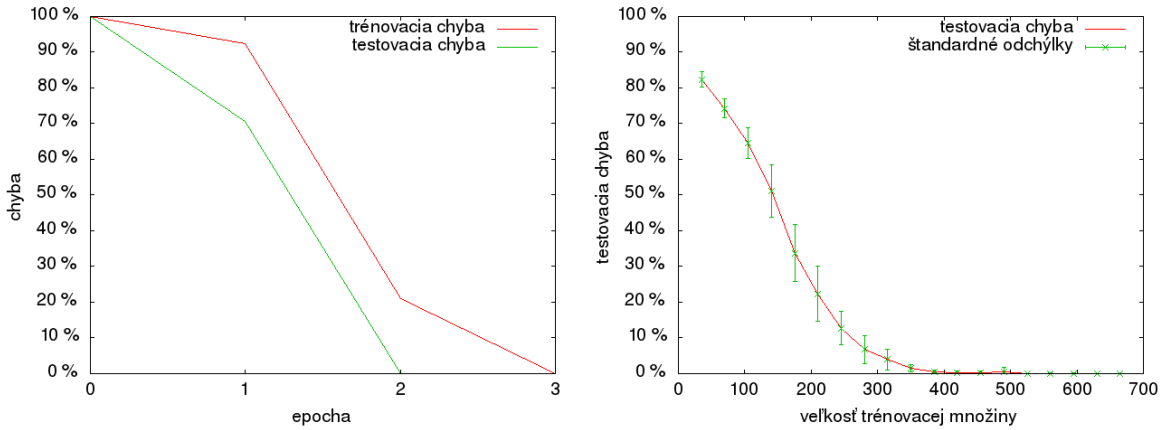
Parametre algoritmu spätného šírenia chyby boli: rýchlosť učenia $\alpha = 0.1$ a moment $\mu = 0.7$. Ostatné parametre (veľkosť skrytej vrstvy, rýchlosť učenia *PCA* a pod.) sú uvedené ďalej pri jednotlivých modeloch a sú identické pre model s implicitnou a explicitnou reprezentáciou, aby bolo možné vidieť vplyv reprezentácie vstupu na učenie modelu. Upozorňujeme, že parametre modelov, ktoré sme použili, nemusia byť optimálne.

4.2 Model AB1-I

Pre základný model sme zreprodukovali experimenty na rotačných a kategorických analógiách, ktoré vykonal Blank, aby sme výsledky mohli porovnať. My sme navyše testovali aj **symetrické analógie**. Reprezentácia vstupov bola zväčšená tenzorová (sekcia 2.2.1), čo znamená, že na vstupnej vrstve bolo $49 \times 6 = 294$ neurónov reprezentujúcich celú scénu, 49 neurónov pre určenie pozície označeného objektu a 1 *bias* neurón – spolu 344 neurónov. Skrytá vrstva mala $49+1 = 50$ neurónov a výstupná vrstva mala $49+49 = 98$ neurónov. Model sme trénovali tak, ako to bolo opísané v časti 3.1.3.

4.2.1 Rotačné analógie

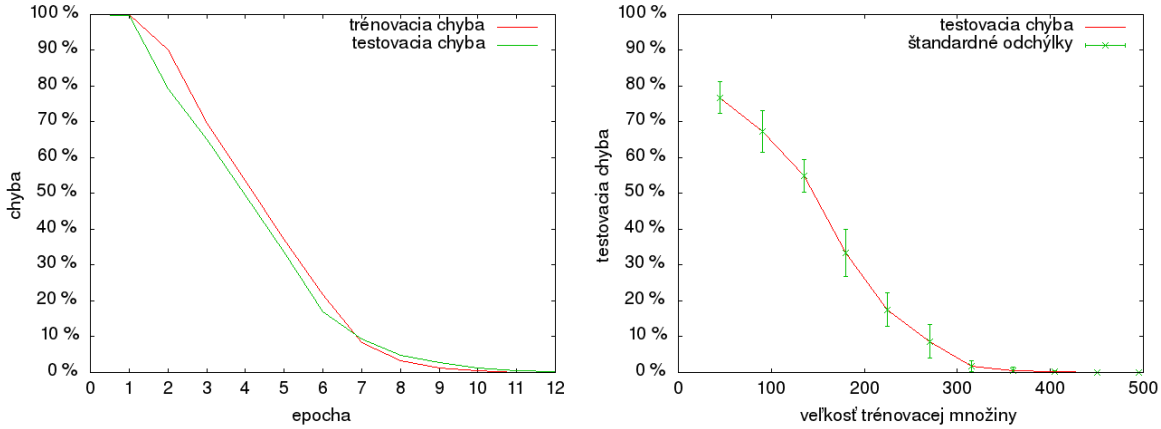
Základný model sa dokázal naučiť rotačné analógie bez akýchkoľvek problémov. Experiment mal ukázať, či sa model naučí zameriavať sa práve na pozíciu objektov a ich črty bude ignorovať. Blank tvrdí, že jeho model potreboval 11800 trénovacích vzorov pre nulovú testovaciu chybu, čo je v súlade s našimi výsledkami, pri ktorých model pracoval s množinou vzorov o veľkosti 3888 a potreboval 3 epochy, aby sa problém naučil. Ďalej sme skúsili zistiť koľko skutočne vzorov model potrebuje, aby dával nulovú testovaciu chybu a vyšlo to na približne 500 trénovacích vzorov (obrázok 4.1).



Obr. 4.1: Správanie modelu AB1-I na rotačných analógiach. Vľavo: Výsledok učenia jednej inštancie modelu AB1-I. Vpravo: Vplyv veľkosti trénovacej množiny na testovaciu chybu.

4.2.2 Symetrické analógie

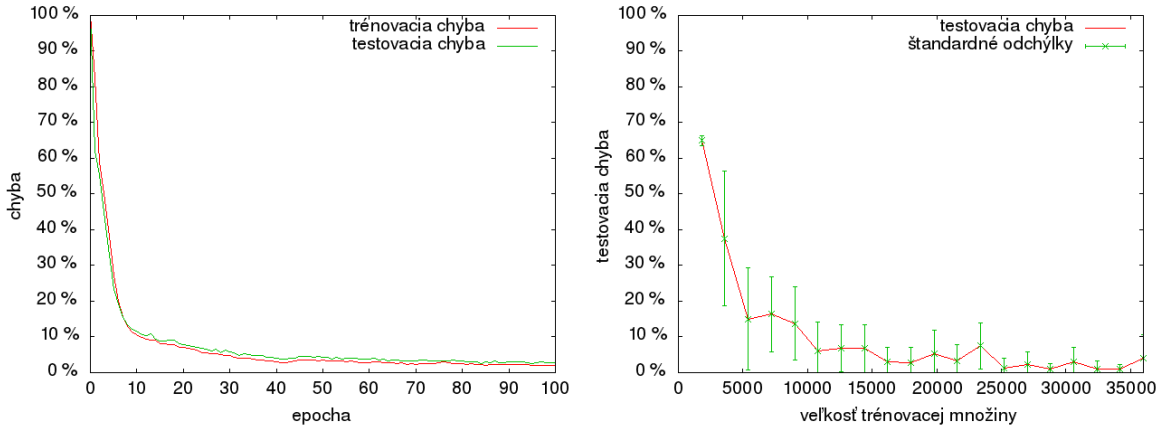
Tento typ analógie Blank netestoval, no my sme sa rozhodli skúsiť to, aby sme videli ako si základný model poradí aj s týmto typom analógie. Ide o podobný typ, ako sú rotácie, pri ktorých nie sú podstatné črty, ale iba pozície objektov. Vygenerovali sme všetky symetrické analógie a náhodne sme vybrali 600 vzorov. Simulácie ukázali, že sieť potrebuje dvanásť epoch na to, aby sa dokázala naučiť sa symetrické analógie. Veľkosť trénovacej množiny potrebnej na naučenie začína pri hodnote 500, čo je pozoruhodné, pretože celá množina analogických dvojíc má veľkosť 26244, no len cca 1.9% tejto množiny je potrebných na to, aby testovacia chyba bola nulová. Správanie modelu je na obrázkoch 4.2.



Obr. 4.2: Správanie modelu AB1-I na symetrických analógiach. Vľavo: Výsledok učenia jednej inštancie modelu AB1-I. Vpravo: Vplyv veľkosti trénovacej množiny na testovaciu chybu.

4.2.3 Kategorické analógie

V tomto experimente Blank zisťoval, či sa model dokáže naučiť analógie, pri ktorých sú dôležité samotné črty a nie pozícia. Blank uvádza, že v jeho experimentoch sa model dokázal problém naučiť, aj keď naše experimenty ukazujú niečo iné. Zistili sme, že *niektorý* model sa dokáže naučiť kategorické analógie a iný zase nie. Blank si zrejme vybral iba jednu inštanciu modelu, ktorá mu dávala dobré výsledky a jej správanie zverejnil vo svojej práci. Usudzujeme to na základe toho, že Blank nerobil priemer simulácií tak ako my – nikde explicitne nenapísal, že ho robil.



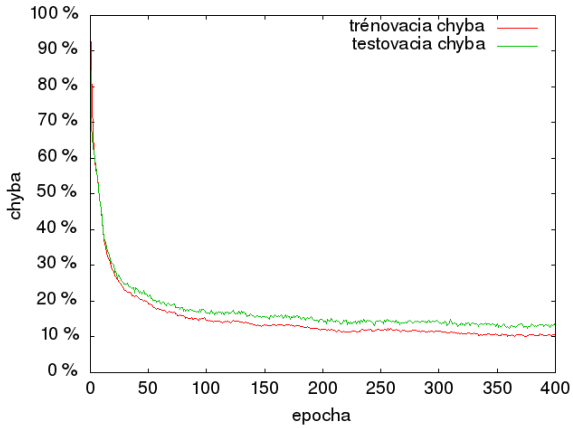
Obr. 4.3: Správanie modelu AB1-I na kategorických analógiách. Vľavo: Výsledok učenia jednej inštancie modelu AB1-I. Vpravo: Vplyv veľkosti trénovacej množiny na testovaciu chybu.

Blank tvrdí, že dosiahol nulovú testovaciu chybu približne pri piatej epoche (uvádza, že po 100000. iterácií, čo je v prepočte päť epoch o veľkosti 20000, s ktorou sme pracovali my). Priemer našich simulácií (obrázok 4.3) je blízky hodnote 3% po 100. epoche, pri ktorej vyzerá, že klesanie chyby sa ustálilo, čo znamená, že vo všeobecnosti sa model nedokáže naučiť kategorické analógie, ako uvádzal Blank. Skúsili sme viacero veľkostí trénovacej množiny, no ani pri veľkosti 36000 sa model problém nenaučil.

4.2.4 Kategorické analógie + rotačné analógie

V tomto experimente sme skombinovali dva typy analógie (kategorické a rotačné analógie) za účelom zistiť, ako veľmi zlé bude správanie, aby sme to mohli porovnať s modifikovanými modelmi. Pretože model sa nedokáže naučiť ani kategorické analógie, je zrejmé, že chyby pri učení budú vyššie.

Model sa problém nedokázal naučiť aj napriek veľkému množstvu epoch (v porovnaní s ďalšími modelmi). Chyba, ktorú dával, bola 12.6%, čo je približne 310 nesprávne určených analógií. Vplyv veľkosti trénovacej množiny sme netestovali, nakoľko by to bolo zbytočné. Na obrázku 4.4 je znázornený graf učenia sa modelu.



Obr. 4.4: Výsledok učenia jednej inštancie modelu AB1-I na kombinácií kategorických a rotačných analógiách.

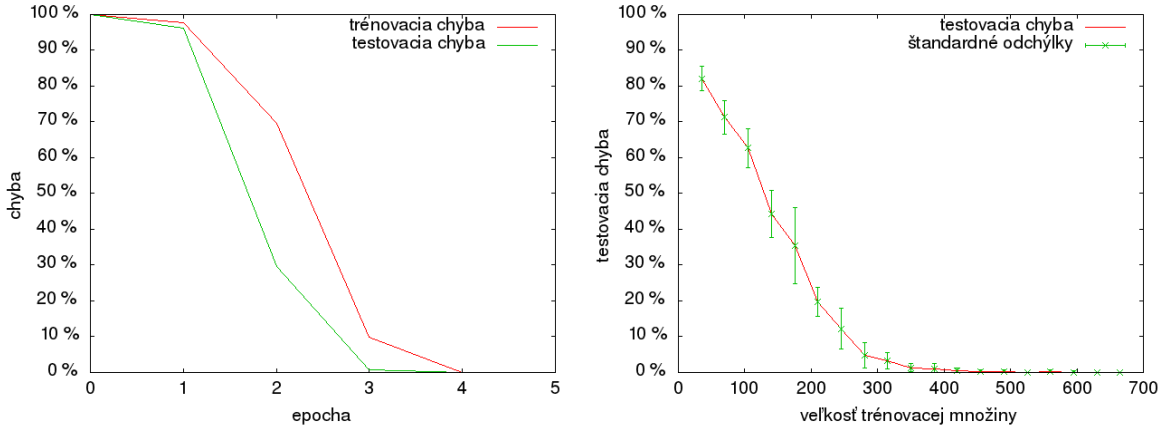
4.3 Model AB1-E

Model AB1-E používa explicitnú reprezentáciu vstupov. Vykonali sme rovnaké experimenty, ako pri modeli AB1-I, aby sme mohli vidieť akú zmenu priniesla nová reprezentácia. Zakódovanie celej scény vyžaduje 24 neurónov a pozície 49 neurónov. Vstupná scéna mala

teda veľkosť 24+49+1=74 neurónov. Skrytá a výstupná vrstva boli rovnako veľké ako pri modeli AB1-I – 50 neurónov na skrytej vrstve a 98 neurónov na výstupnej vrstve. Všetky ostatné parametre (okrem reprezentácie vstupu) boli rovnaké ako pri modeli AB1-I.

4.3.1 Rotačné analógie

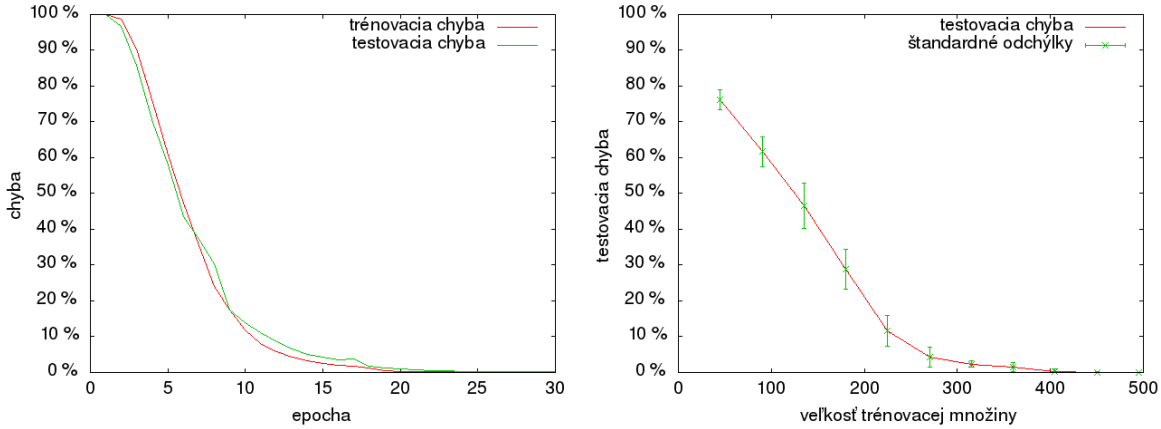
Ako aj v prípade AB1-I modelu, použili sme množinu vzorov o veľkosti 3888. Model potreboval v priemere o jednu epochu viac ako potreboval model AB1-I. Čo sa týka veľkosti trénovacej množiny potrebnej pre naučenie modelu, je približne rovnaká ako pri modeli AB1-I. Zmena reprezentácie vstupu zrejme nerobí pri tomto type analógie podstatný rozdiel.



Obr. 4.5: Správanie modelu AB1-E na rotačných analógiach. Vľavo: Výsledok učenia jednej inštancie modelu AB1-E. Vpravo: Vplyv veľkosti trénovacej množiny na testovaciu chybu.

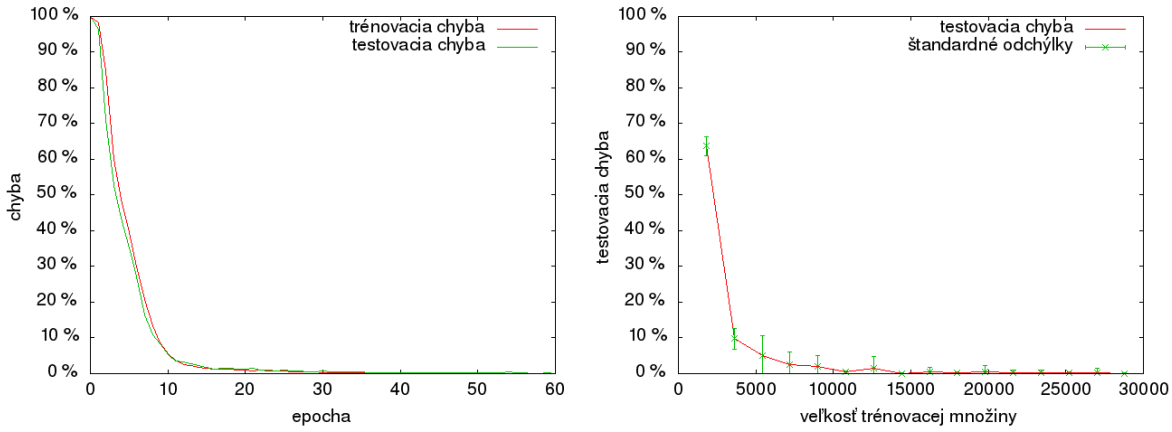
4.3.2 Symetrické analógie

V tomto experimente už vidíme pomerne veľký rozdiel medzi implicitnou a explicitnou reprezentáciou. V priemere sa model naučil problém za 30 epoch, no bola tu tendencia uviaznutia v lokálnom minime, v ktorom model pobudol niekoľko epoch, kým sa dostal von, a ktorú model AB1-I nemal. Pri modeli s implicitnou reprezentáciou sme si nikdy nevšimli, že by model zapadol do lokálneho minima, no pri modeli AB1-E to je možné. V priemere je model dvojnásobne horší oproti modelu AB1-I, pretože potrebuje dvojnásobný počet epoch. Veľkosť trénovacej množiny potrebná pre naučenie problému je rovnaká ako pri modeli AB1-I. Správanie je na obrázku 4.6.



Obr. 4.6: Správanie modelu AB1-E na symetrických analógiach. Vľavo: Výsledok učenia jednej inštancie modelu AB1-E. Vpravo: Vplyv veľkosti trénovacej množiny na testovaciu chybu.

4.3.3 Kategorické analógie



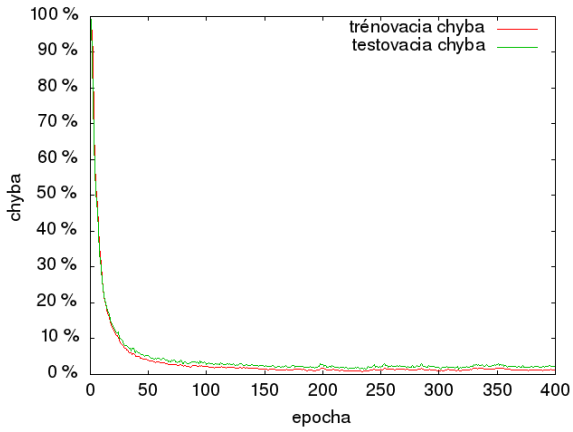
Obr. 4.7: Správanie modelu AB1-E na kategorických analógiách. Vľavo: Výsledok učenia jednej inštancie modelu AB1-E. Vpravo: Vplyv veľkosti trénovacej množiny na testovaciu chybu.

Aj pri tomto experimente je vidieť jasný rozdiel medzi použitou reprezentáciou. Zatiaľ čo každá inštancia modelu AB1-I sa nedokázala naučiť kategorické analógie, pri modeli sa to naopak dokáže každá. Explicitná reprezentácia podstatne pomáha pri riešení problému a umožňuje každej inštancii modelu naučiť sa kategorické analógie.

Obrázok 4.7 zobrazuje, že model potrebuje trénovaciu množinu o veľkosti 15000, aby sa úspešne dokázal problém naučiť. Veľkosť, pri ktorej sme učili model, bola 18000. Pri nej sa model naučil problém za 30 epoch, ale potreboval ešte ďalších 30 epoch, aby minimalizoval

testovaciu chybu na nulu.

4.3.4 Kategorické analógie + rotačné analógie



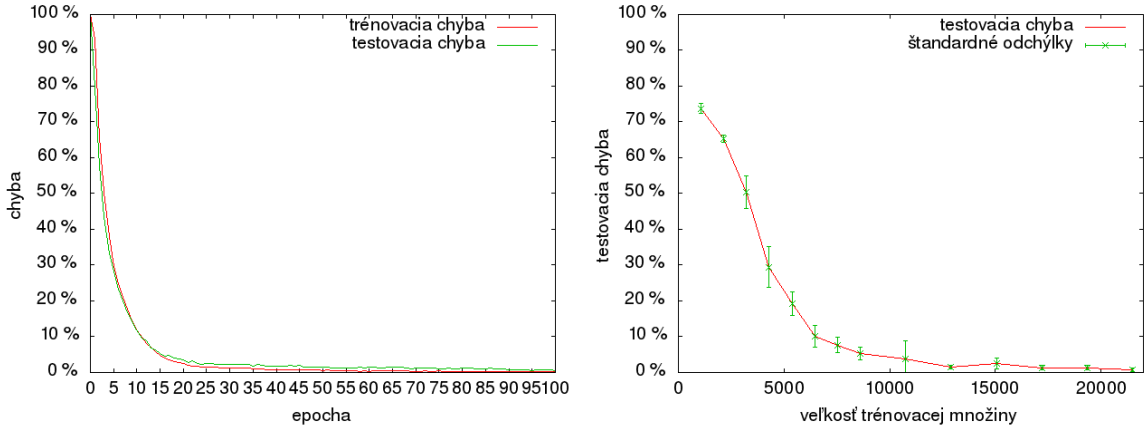
Obr. 4.8: Výsledok učenia jednej inštancie modelu AB1-E na kombinácií kategorických a rotačných analógiách.

Obrázok 4.8 zobrazuje podstatné zlepšenie, aj keď stále sa nedá povedať, že model sa dokázal úplne naučiť oba typy analógie. Testovacia chyba je už 2.2%, čo je zlepšenie približne o 10% oproti modelu AB1-I, a čo vychádza približne na 54 nesprávne určených analógií z 2388 (veľkosť testovacej množiny). Model zároveň potrebuje podstatne menší počet epoch, aby skonvergoval (našiel minimálnu chybu), a potrebuje menej času na jednu iteráciu (kvôli menšiemu vstupu).

4.4 Model AP1-I

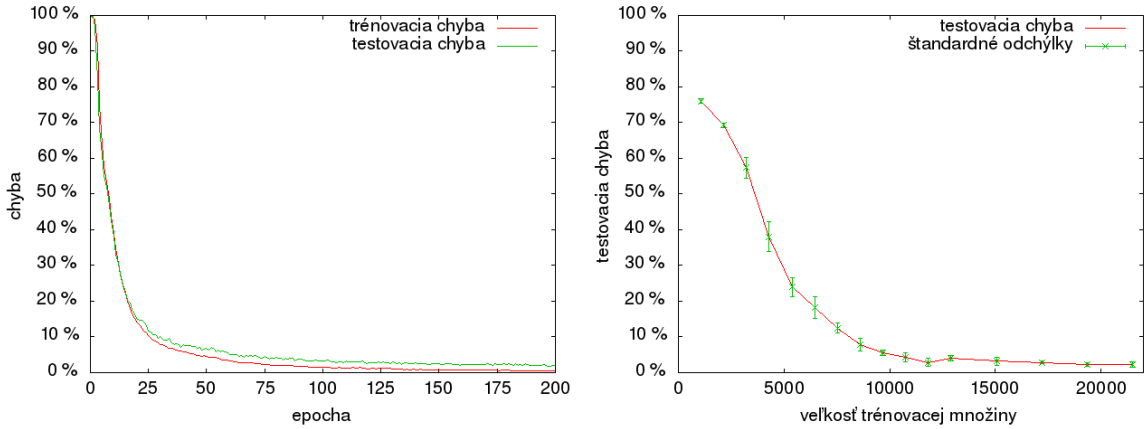
Model AP1-I používa *PCA* na transformáciu skrytej vrstvy do časti vstupnej vrstvy a implicitnú, zväčšenú tenzorovú, reprezentáciu. Na skrytú vrstvu sme umiestnili 100 neurónov a rýchlosť učenia *PCA* sme nastavili na $\alpha = 0.0001$. Vyššie hodnoty α sa ukázali byť kontraproduktívne a pri hodnote $\alpha = 0.1$ sa model už nedokázal vôbec niečo naučiť. Trénovacia chyba v takomto prípade nikdy nezliezla pod 100%. Skúsili sme aj 200 neurónov na skrytej vrstve, ale učenie to nevylepšilo, preto takúto simuláciu tu neuvádzame.

Model sa naučil kombináciu kategorických a rotačných analógii niekde pri 75. epoche a pri 100. epoche dosiahol testovaciu chybu 0.6%, čo je 14.3 nesprávne určených analógií. Optimálna veľkosť trénovacej množiny sa javila byť maximálna zo všetkých, ktoré sme skúsili – 21500. Správanie modelu je zachytené na obrázkoch 4.9.



Obr. 4.9: Správanie modelu AP1-I na kombinácií kategorických a rotačných analógiách. Vľavo: Výsledok učenia jednej inštancie modelu AP1-I. Vpravo: Vplyv veľkosti trénovacej množiny na testovaciu chybu.

4.5 Model AP1-E

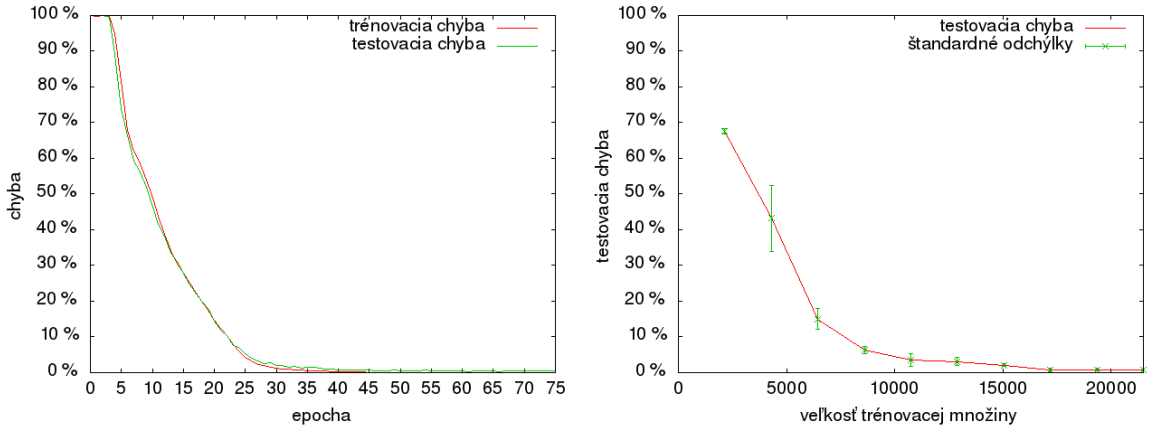


Obr. 4.10: Správanie modelu AP1-E na kombinácií kategorických a rotačných analógiách. Vľavo: Výsledok učenia jednej inštancie modelu AP1-E. Vpravo: Vplyv veľkosti trénovacej množiny na testovaciu chybu.

Explicitná reprezentácia pomohla pri modeli AB1-E, tak je otázne, ako bude pracovať s týmto modelom. V porovnaní s modelom AP1-I, model AP1-E dáva vyššiu testovaciu chybu a potrebuje vyšší počet epoch, aby skonvergoval (dával minimálnu testovaciu chybu). Testovacia chyba je trojnásobne vyššia ako pri modeli AP1-I – 2.0%, čo vychádza na 47.76 nesprávne vytvorených analógií. Vplyv rôznych veľkostí vyzerá byť rovnaký ako pri AB1-I. Správanie modelu je zachytené na obrázkoch 4.10. Explicitná reprezentácia teda nemá pozitívny prínos pri modeli AP1-E.

4.6 Model AS1-I

Tento model má oddelenú časť vstupnej vrstvy, ktorá určuje pozíciu označeného objektu a zároveň hrá úlohu kontextových neurónov. Vstupy boli reprezentované implicitne a bola použitá klasická tenzorová reprezentácia, nie zväčšená. Na skrytú vrstvu sme umiestnili 100+1 neurónov, tým pádom vstupná vrstva mala $4\times6+4+100+1 = 129$ neurónov. Použitá množina množiny vzorov mala veľkosť 23888.

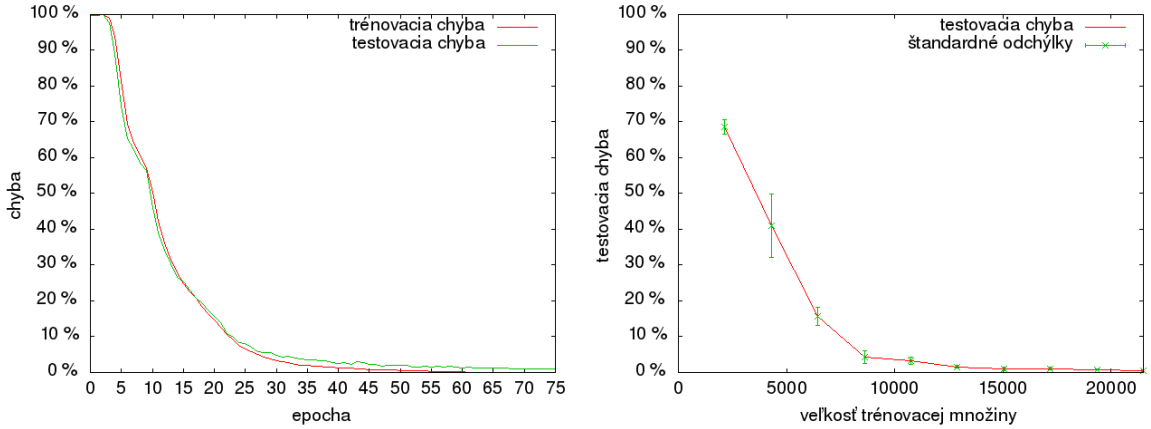


Obr. 4.11: Správanie modelu AS1-I na kombinácií kategorických a rotačných analógiách. Vľavo: Výsledok učenia jednej inštancie modelu AS1-I. Vpravo: Vplyv veľkosti trénovacej množiny na testovaciu chybu.

Simulácie ukázali, že tento model sa vie úspešne naučiť oba typy analógií s menším počtom epoch, ale testovacia chyba nie je stále nulová. Po 75. epoche model skonvergoval ku testovacej chybe 0.4% - 9.6 nesprávne určených analógií. Priebeh simulácií je znázornený na obrázku 4.11.

4.7 Model AS1-E

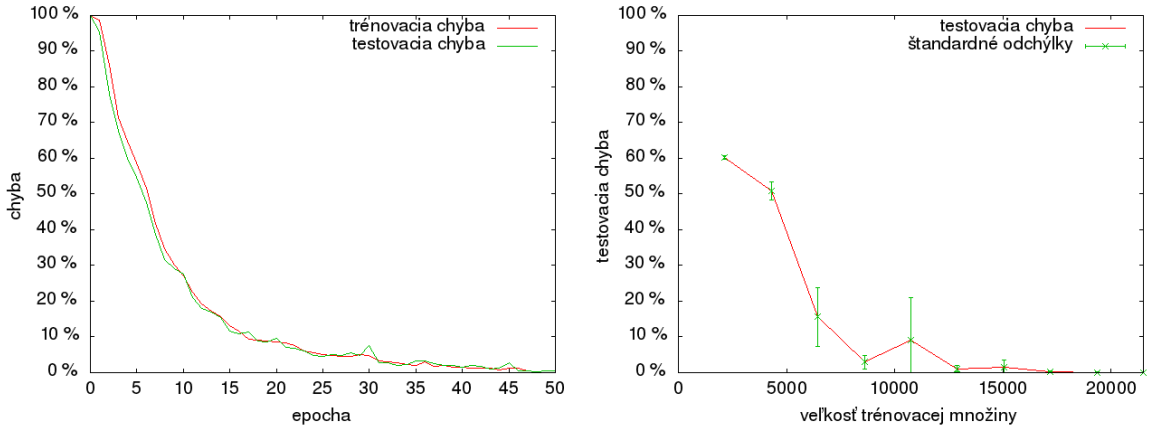
Pretože explicitná reprezentácia mala prínos pri základnom modeli, otázka je či bude mať prínos aj pri tomto modeli. Všetky parametre sme nechali rovnaké ako pri modeli s implicitnou reprezentáciou - nezväčšená reprezentácia a 100 neurónov na skrytej vrstve. Graf simulácií (obrázok 4.12) ukazuje veľmi slabé zhoršenie. Model konverguje približne rovnako rýchlo a pri 75. epoche dáva testovaciu chybu 1.0% – 23.8 nesprávne určených analógií. Použitá reprezentácia pri tomto modeli zrejme nie je až taká podstatná.



Obr. 4.12: Správanie modelu AS1-E na kombinácií kategorických a rotačných analógiách. Vľavo: Výsledok učenia jednej inštancie modelu AS1-E. Vpravo: Vplyv veľkosti trénovacej množiny na testovaciu chybu.

4.8 Model AB2-I

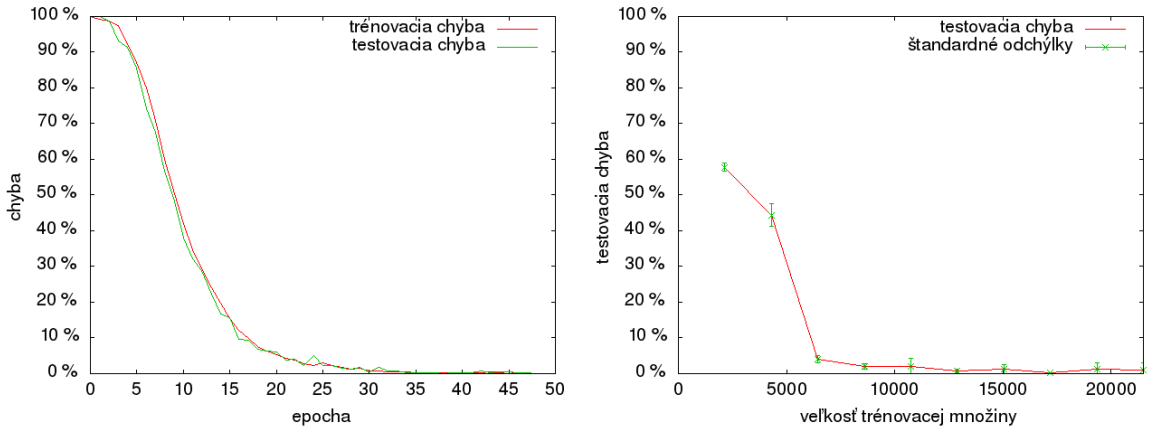
Model obsahuje dodatočnú skrytú vrstvu, na ktorú sme umiestnili 100+1 skrytých neurónov. Vstupná vrstva a pôvodná skrytá vrstva má rovnakú veľkosť ako mal model AB1-I. Použitá množina množiny vzorov mala veľkosť 23888. Reprezentácia vstupov bola zväčšená tenzorová.



Obr. 4.13: Správanie modelu AB2-I na kombinácií kategorických a rotačných analógiách. Vľavo: Výsledok učenia jednej inštancie modelu AB2-I. Vpravo: Vplyv veľkosti trénovacej množiny na testovaciu chybu.

Model sa naučil problém približne pri 50. epoche a optimálna veľkosť trénovacej množiny sa javí byť 19000. Testovacia chyba, ktorú dával bola 0.4% – 9.5 nesprávne vytvorených analógií. Správanie modelu je zachytené na obrázkoch 4.13.

4.9 Model AB2-E



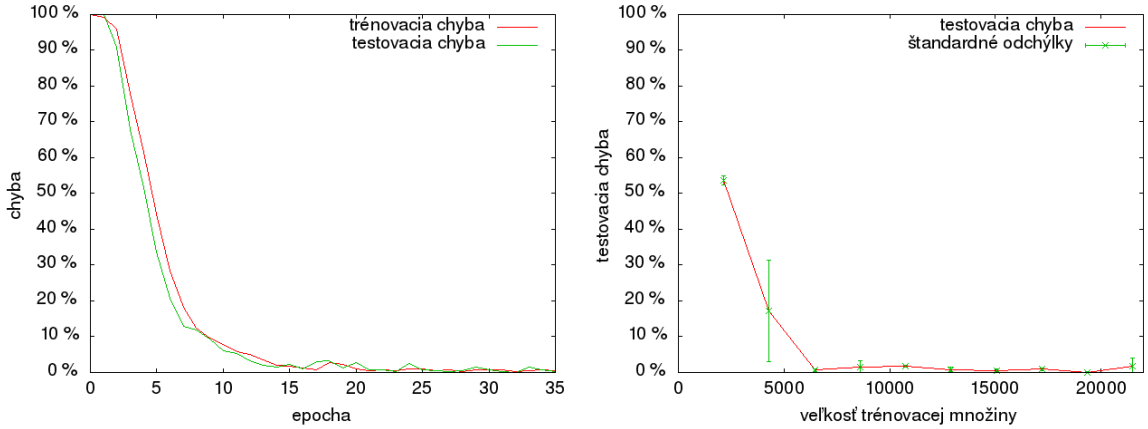
Obr. 4.14: Správanie modelu AB2-E na kombinácií kategorických a rotačných analógiách. Vľavo: Výsledok učenia jednej inštancie modelu AB2-E. Vpravo: Vplyv veľkosti trénovacej množiny na testovaciu chybu.

Explicitná reprezentácia pri tomto modeli má prínos a model sa tak dokáže problém naučiť rýchlejšie ako model AB2-I. Model sa naučil problém už pri 35. epoche a optimálna veľkosť trénovacej množiny začína pri hodnote 13000. Testovaciu chybu, ktorú model urobil po 50. epoche, bola 0.1% – 2.4 zle určených analógií. Grafy simulácií sú zachytené na obrázkoch 4.14.

4.10 Model AP2-I

Tento model používal lineárnu transformáciu pôvodnej skrytej vrstvy do časti vstupnej vrstvy pre určenie pozície označeného objektu pomocou *PCA* a ďalšiu skrytú vrstvu. Na obe skryté vrstvy sme umiestnili 100+1 neurónov a rýchlosť učenia pre *PCA* sme nastavili na $\alpha = 0.0001$. Reprezentácia vstupov bola rozšírená tenzorová.

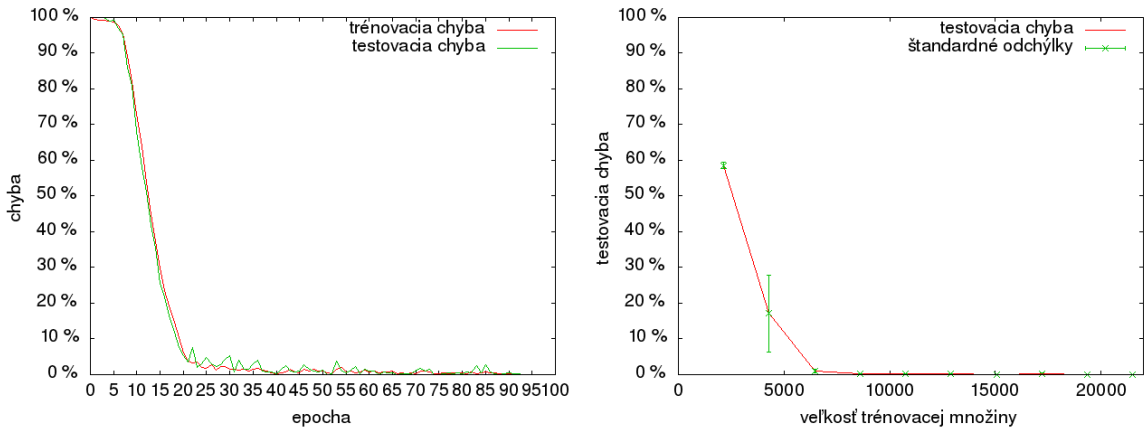
Model dosiahol testovaciu chybu 0.14% pri 35. epoche, čo je v prepočte 3.34 chybne vytvorených analógií. Optimálna veľkosť trénovacej množiny začína už pri hodnote 7500, čo je obrovský pokles oproti minulým modelom. Správanie modelu je možné vidieť na obrázkoch 4.15.



Obr. 4.15: Správanie modelu AP2-I na kombinácií kategorických a rotačných analógiách. Vľavo: Výsledok učenia jednej inštancie modelu AP2-I. Vpravo: Vplyv veľkosti trénovacej množiny na testovaciu chybu.

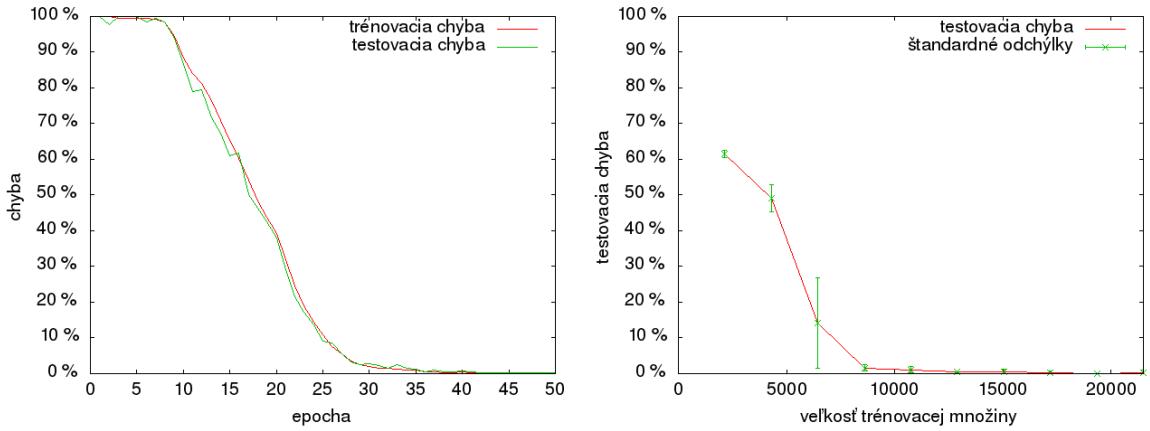
4.11 Model AP2-E

Keďže model AP1-X pracuje lepšie s implicitnou reprezentáciou a model AB2-X má lepšie výsledky pri explicitnej reprezentácií, je zaujímave dozvedieť sa, či bude fungovať lepšie model AP2-E, alebo model AP2-I. Výsledky simulácií ukazujú, že model je lepší v testovacej chybe po naučení, ale potrebuje na to podstatne väčší počet epoch. Po 100. epoche model dáva testovaciu chybu 0.0%, čo je absolútne zovšeobecnenie problému na testovacích dátach. Graf s vplyvom veľkosti na testovaciu chybu toto správanie potvrdzuje a ukazuje, že model potrebuje iba 8500 trénovacích vzorov, aby sa problém úplne naučil. Graf učení sa modelu je zobrazený na obrázkoch 4.16.



Obr. 4.16: Správanie modelu AP2-E na kombinácií kategorických a rotačných analógiách. Vľavo: Výsledok učenia jednej inštancie modelu AP2-E. Vpravo: Vplyv veľkosti trénovacej množiny na testovaciu chybu.

4.12 Model AS2-I



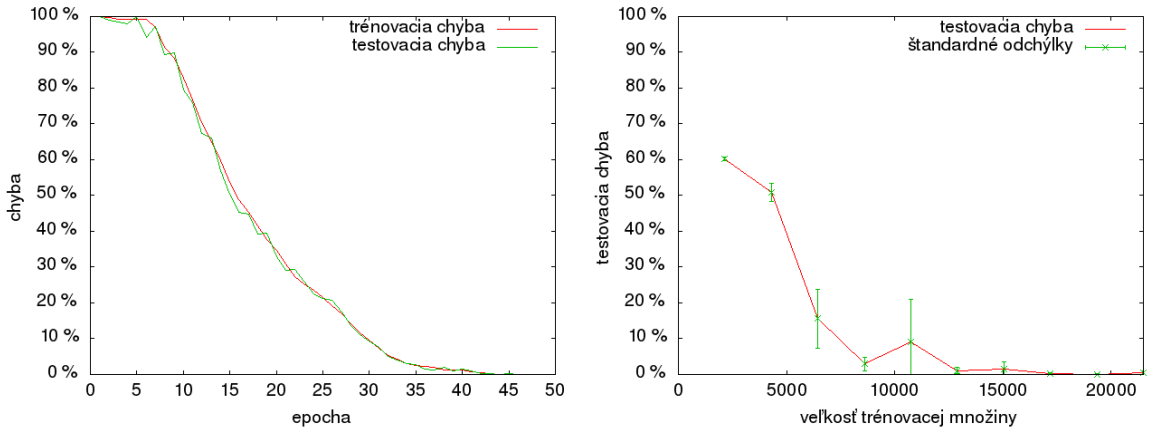
Obr. 4.17: Správanie modelu AS2-I na kombinácií kategorických a rotačných analógiách. Vľavo: Výsledok učenia jednej inštancie modelu AS2-I. Vpravo: Vplyv veľkosti trénovacej množiny na testovaciu chybu.

Model AS2-I sa líši od základnej verzie AB1-I tým, že má oddelenú vstupnú vstupnú vrstvu od kontextových neurónov a obsahuje ďalšiu skrytú vrstvu. Reprezentácia vstupov je základná implicitná, nie zväčšená. Na prvej skrytej vrstve bolo 50 neurónov (čo je aj počet kontextových neurónov na vstupnej vrstve) a na druhej skrytej vrstve bolo takisto 50 neurónov. Použitá množina vzorov mala veľkosť 23888.

Model sa dokázal naučiť oba typy analógie za menej ako 45 epoch s priemernou testovacou chybou 0.19% – 4.5 nesprávne určených analógií. Optimálna veľkosť trénovacej postačuje byť 13000. Správanie modelu je zachytené na obrázkoch 4.17.

4.13 Model AS2-E

Nakoniec posledný model AS2-E, pri ktorom nie je takisto vidieť významný rozdiel oproti modelu AS2-I, podobne ako medzi modelmi AS1-I a AS1-E, aj keď model AS2-I vyzerá byť o čosi lepší. Model AS2-E potrebuje 45 epoch pre naučenie a optimálna veľkosť trénovacej množiny začína pri hodnote 17000. Testovaciu chybu dáva 0.04% – 0.95 chybne určenej analógie. Správanie modelu je zachytené na obrázkoch 4.18.



Obr. 4.18: Správanie modelu AS2-E na kombinácií kategorických a rotačných analógiách. Vľavo: Výsledok učenia jednej inštancie modelu AS2-E. Vpravo: Vplyv veľkosti trénovacej množiny na testovaciu chybu.

4.14 Zhrnutie výsledkov

Vykonali sme experimenty s rotačnými, symetrickými, kategorickými analógiami a zároveň sme skúsili kombináciu kategorických a rotačných analógií. Jeden typ analógie sa vedeli naučiť všetky modely, vrátane základných modelov AB1-X. Model AB1-I sa vo väčšine prípadov vedel naučiť kategorické analógie, no nie každej inštancii sa to podarilo. Model AB1-E s tým už problém nemal.

Naučiť sa dva typy analógie (rotačné a kategorické analógie) vedeli všetky modely okrem základných, modelu AP1-E, ktorý dával 2.0% testovaciu chybu po 200. epoche, a modelu AS1-E, ktorý dával 1.0% testovaciu chybu po 75. epoche. Ostatné modely dávali chybu menšiu ako 1.0%.

Najlepší model, ktorý sme objavili, je model AP2-E, ktorý používal 100 neurónov na oboch skrytých vrstvách. Každá inštancia modelu sa vedela naučiť oba typy analógie s nulovou testovacou chybou, pričom modelu vystačí iba 8500 trénovacích dát, aby minimalizoval testovaciu chybu. Ostatné modely potrebovali približne dvojnásobnú veľkosť trénovacej množiny.

Záver

V práci sme sa venovali problému geometrických analógií, ktoré spadajú pod analógie typu *časť–celok*. Zreprodukovali sme experimenty, ktoré vykonal Blank a skúsili sme aj vlastné experimenty, na ktorých model Analogator už nepracoval ideálne. Preto sme navrhli modifikácie, ktoré sa ukázali byť prínosom pre model, a ktoré zlepšili správanie modelu aj v našich experimentoch. Takisto sme skúsili použiť aj explicitnú reprezentáciu, namiesto Blankovej implicitnej, ktorá bola lepšia v niektorých modeloch, aj keď nie vo všetkých.

Spolu sme vyhodnotili správanie základnej verzie modelu a piatich našich modelov. Najlepší model, ktorý sme našli, je model rozšírený o metódu hlavných komponentov (*PCA*), ktorý pracuje s implicitnou reprezentáciou. Tento model sa dokázal najrýchlejšie učiť (čo sa týka počtu epoch) a každá inštancia modelu dávala vždy nulovú trénovaciu chybu a takmer nulovú testovaciu chybu. Naša explicitná reprezentácia pomáha modelom efektívnejšie sa učiť (okrem modelu s *PCA*), aj keď niektorí kognitívni vedci by nesúhlasili s používaním explicitných reprezentácií.

Čo sme nepreskúmali je vplyv rôznych aktivačných funkcií (použili sme iba unipolárnu sigmoidu), rôznych rýchlosti učenia a rôznych momentov. Predpokladali sme, že Blank našiel ideálne hodnoty týchto parametrov, aj keď nie je vôbec isté, či sú ideálne aj v našich modeloch. Ďalší vývoj by mohol smerovať pri vyhodnotení modelu spolu s našimi modifikáciami na iných problémoch, pri ktorých by sa mohlo bližšie objasniť ako veľmi zásadné sú naše modifikácie.

Zoznam literatúry

Blank, D. S. (1997). *Learning to See Analogies: A Connectionist Exploration*. Dizertačná práca, Indiana University, Bloomington.

Chalmers, D. J., French, R. M., and Hofstadter, D. R. (1992). High-level perception, representation, and analogy: A critique of artificial intelligence methodology. *Journal of Experimental and Theoretical Artificial Intelligence*, 4:185–211.

Duncker, K. and Lees, L. (1945). *On problem-solving*. Psychological monographs. The American Psychological Association, inc.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14(2):179–211.

Evans, T. G. (1968). A program for the solution of a class of geometric-analogy intelligence-test questions. In Minsky, M. L., editor, *Semantic Information Processing*, pages 271–353. MIT Press, Cambridge, Massachusetts.

Falkenhainer, B., Forbus, K. D., and Gentner, D. (1986). The structure-mapping engine. In *Proceedings of the 5th National Conference on Artificial Intelligence*, pages 272–277, Philadelphia, PA.

French, R. and Hofstadter, D. (1992). Probing the emergent behavior of tabletop, an architecture uniting high-level perception with analogy-making. In *Proceedings of the 14th Annual Cognitive Science Society Conference*, pages 183–193.

French, R. M. (2002). The computational modeling of analogy-making. *Trends in Cognitive Sciences*, 6(5):200–205.

Gentner, D. (1982). *Are Scientific Analogies Metaphors?* Harvester Press Ltd.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155–170.

Gentner, D. and Bolt, B. (1980). *The Structure of Analogical Models in Science*. Distributed by ERIC Clearinghouse [Washington, D.C.].

Gick, M. L. and Holyoak, K. J. (1980). Analogical problem solving. *Cognitive Psychology*, 12(3):306–355.

Gick, M. L. and Holyoak, K. J. (1983). Schema induction and analogical transfer. *Cognitive Psychology*, 15(1):1–38.

Goswami, U. (1991). Analogical reasoning: What develops? a review of research and theory. *Child Development*, 62:1–22.

Halford, S. G., Wilson, W., Guo, J., Gaylor, R. W., Wiles, J., and Stewart, J. E. M. (1994). Connectionist implications for processing capacity limitations in analogies. In *Advances in Connectionist and Neural Computation Theory, Volume 2: Analogical Connections*, pages 363–445. Ablex Publishing, Norwood, New Jersey.

Hall, R. P. (1989). Computational approaches to analogical reasoning: A comparative analysis. *Artificial Intelligence*, 39:39–120.

Holcombe, A. O. (2009). The binding problem. In *Encyclopedia of Perception*. SAGE Publications, Inc.

Holyoak, K. J. and Thagard, P. (1989). Analogical mapping by constraint satisfaction. *Cognitive Science*, 13:295–355.

Hummel, J. E. and Holyoak, K. J. (2005). Relational reasoning in a neurally plausible cognitive architecture: An overview of the LISA project. *Current Directions in Psychological Science*, 14(3):153–157.

Jani, N. G. and Levine, D. S. (2000). A neural network theory of proportional analogy-making. *Neural Networks*, 13(2):149–183.

Kosslyn, S. M., Koenig, O., Barrett, A., Cave, C. B., Tang, J., and Gabrieli, J. D. (1989). Evidence for two types of spatial representations: hemispheric specialization for categorical and coordinate relations. *Journal of Experimental Psychology: Human Perception and Performance*, 15(4):723–35.

Kung, S. Y. and Diamantaras, K. I. (1990). A Neural Network Learning Algorithm for Adaptive Principal Component EXtraction (APEX). In *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Processing*, pages 861–864.

Landau, B. and Jackendoff, R. (1993). "What" and "Where" in spatial language and spatial cognition. *Behavioral and Brain Sciences*, 16:217–238.

Mitchell, M. (2001). Analogy-making as a complex adaptive system.

Mitchell, M. and Hofstadter, D. (1995). Perspectives on copycat: Comparisons with recent work. In *Fluid Concepts and Creative Analogies*, pages 275–299. Basic Books, Inc., New York, NY.

Mithen, S. (1996). *The Prehistory of the Mind: A Search for the Origins of Art, Religion and Science*. Thames and Hudson.

Oja, E. (1982). Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15:267–273.

Rao, S. C., Rainer, G., and Miller, E. K. (1997). Integration of what and where in the primate prefrontal cortex. *Science*, 276(5313):821–824.

Sanger, T. D. (1989). Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2:459–473.

Thompson, R. K., Oden, D. L., and Boysen, S. T. (1997). Language-naive chimpanzees (Pan troglodytes) judge relations between relations in a conceptual matching-to-sample task. *Journal of Experimental Psychology: Animal Behavior Processes*, 23(1):31–43.

Ungerleider, L. G. and Mishkin, M. (1982). Two cortical visual systems. In Ingle, D. J., Goodale, M. A., and Mansfield, R. J. W., editors, *The Analysis of Visual Behavior*, pages 549–586. MIT Press, Cambridge, MA.

Príloha A

Ku práci prikladáme CD médium, ktoré obsahuje:

- implementáciu všetkých modelov opísaných v tejto práci v programovacom jazyku Java (verzia 1.7)
- dokumentáciu zdrojového kódu v anglickom jazyku
- kompletné výsledky z experimentov uložené v textových súboroch
- bash skripty na vytvorenie grafov z výsledkov experimentov pomocou programu *gnuplot*
- samotnú diplomovú prácu
- grafy a použité obrázky v anglickom jazyku