Študentská vedecká konferencia FMFI UK, Bratislava, 2014, pp. 60–68
ISBN 978-80-8147-023-3, © 2014 Peter Gergeľ

Konekcionistické modelovanie učenia sa analógií

Peter Gergeľ*

Školiteľ: Igor Farkaš†

Katedra aplikovanej informatiky, FMFI UK, Mlynská Dolina 842 48 Bratislava

Abstrakt: V práci prinášame nové metódy v oblasti modelovania učenia sa analógií, ktoré zlepšujú učenie sa oproti existujúcim modelom. Model *Analogator*, z ktorého vychádzame, mal obmedzenia, ktoré mu síce umožnili naučiť sa časť problému, no nedovolili mu naučiť sa celý problém. Naše modifikácie modelu odstraňujú toto obmedzenie, vďaka čomu potenciál modelu narástol. Oblasť analogického modelovania sa tak môže rozšíriť o naše nové poznatky.

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

1 Úvod

Analogické uvažovanie je fundamentálna kognitívna schopnosť ľudí, ktorou sa človek líši od väčšiny zvierat. 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 podať hypotézy a vysvetlenia.

Vytvoriť analógiu znamená vidieť nejaký objekt, alebo situáciu, v jednom kontexte rovnako ako iný objekt, alebo situáciu, v druhom kontexte. Podľa Hall (1989) je tento proces také zobrazenie medzi dvoma doménami nazvanými *zdroj* a *cieľ*, že objekty, ktoré majú rovnaký význam vrámci domény sa 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.

*peter.gergel@gmail.com
†farkas@fmph.uniba.sk

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*). V našej práci sme prvé dva kroky preskočili a venujeme sa len modelovaniu *prenosu znalostí zo zdroja na cieľ* a *konsolidácii*, nakoľko v povahe problému, ktorý sme si vybrali, je už zdroj zadaný.

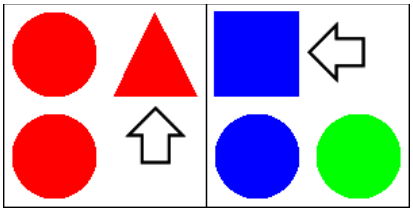
Na úvod je potrebné ešte vysvetliť aké analógie existujú a ktorým typom sa v práci venujeme. Za klasický typ analógie, ktorý sa datuje do doby Starovekého Grécka, sa považuje *propozičná analógia*, ktorá má tvar *A ku B je ako C ku D*. Patria sem *synonymické analógie* (profesor k učiteľovi je ako študent 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). Ďalší možný typ sú *metafory*, pri ktorých sa povaha analógie nešpecifikuje (Jožo je veľký ako slon). V našej práce sa venujeme typu analógie *časť – celok*.

2 Geometrické analógie

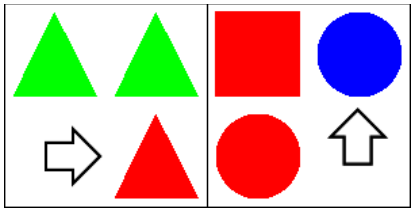
Ako problém, na ktorom chceme vyhodnocovať správanie modelu, sme si vybrali problém geometrických analógií, ktorý použil aj Blank (1997) vo svojej dizertačnej práci, a ktorý sa vyskytuje často aj na IQ testoch. Problém bol zvolený za účelom porovnania dosiahnutých výsledkov a navyše mnohým, vrátane nás, 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



Obr. 1: Príklad geometrickej analógie.



Obr. 2: Príklad farebnej analógie.

označiť objekt v druhej scéne, ktorý sa bude líšiť od svojich ostatných objektov presne v tej iste črte, tak ako sa líši objekt v prvej scéne. Ide o hľadanie analógie medzi geometrickými obrazcami.

Ukážka problému je na obrázku 1, kde ľavá časť obrázku je spomínaná prvá scéna a pravá časť obrázku druhá scéna. Šípka v prvej scéne určuje objekt, ktorý je predmetom 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 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 objekt líšil 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* takýchto geometrických analógií a je na nás, ktorým typom sa budeme venovať. My sme si vybrali tieto typy:

- **farebné analógie** (analogický objekt sa líši farbou od ostatných)
- **tvarové analógie** (analogický objekt sa líši tvarom od ostatných)

Ďalšie typy sme si nevybrali kvôli obmedzeným možnostiam tohoto príspevku.

2.1.2 Farebné analógie

Pri farebných analógiách platí to, že analogický objekt je práve ten, ktorý sa líši od ostatných svojou farbou. V pôvodnej scéne existuje len práve táto jedna črta, ktorá rozlišuje objekt, pričom v cieľovej scéne sú už dve rôzne črty. Okrem inej farby je použitý aj iný tvar niektorého iného objektu. Príklad je na obrázku 2. Dôvod prečo je to tak je ten, že chceme, aby si program v pôvodnej scéne všimol, že podstatná črta je farba a aby dokázal túto vedomosť potom aplikovať na cieľovú scénu. Program by sa mal naučiť, že tvar v týchto farebných analógiách je nepodstatný.

Skúsme sa zamyslieť, koľko môže existovať všetkých takýchto farebných analógií. Pri generovaní zdrojovej scény je dôležité zmeniť farbu a ponechať tvar. Počet možných farieb použitých v jednej scéne[1] je teda 6 a počet tvarov, ktoré možno použiť, je 3. Jeden objekt označíme troma spôsobmi a objekty vieme umiestniť štyrmi spôsobmi. Spolu to vychádza $6\times3\times3\times4 = 216$ scén. Cieľové scény sa generujú rovnako s tým rozdielom, že na celej scéne sú 2 rôzne tvary, pričom označený objekt má tvar spoločný s aspoň jedným objektom. Tu nastávajú prípady, že buď všetky objekty majú rovnaký tvar, alebo prvý neoznačený objekt má iný tvar (2 možnosti ako zmeniť tvar), alebo druhý neoznačený objekt má iný tvar. Spolu máme 5 možností ako modifikovať scénu. Cieľová množina má potom veľkosť $216\times5 = 1080$ a výsledná množina analogických dvojíc má veľkosť $216\times1080 = 233280$. Časť tejto výslednej množiny sme potom použili na trénovanie neurónovej siete.

2.1.3 Tvarové analógie

Tvarové analógie sú takmer identické s farebnými analógiami s jediným rozdielom, a to tým, že analogický objekt sa líši tvarom od ostatných. Ako príklad možno použiť obrázok 1. Veľkosť vygenerovanej množiny je rovnaká ako veľkosť množiny farebných analógií. Znova by sme boli radi, ak by si program vedel „uvedomiť", že farba je v prípade tvarových analógií úplne nepodstatná.

[1] Tri možnosti ako vybrať farbu pre označený objekt a dve možnosti výberu farby ostatných objektov.

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.

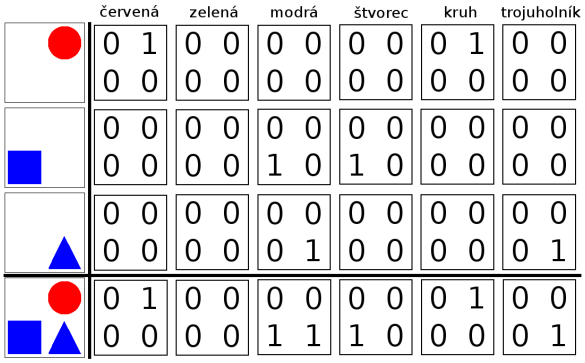
2.2.1 Implicitná reprezentácia

V implicitnej reprezentácii nie sú štruktúry objektov a vzťahy medzi objektami exaktne opísané. 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 analógie podstatne zjednoduší.

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

Blank sa pri vytváraní implicitnej 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** (angl. outer product)[2], 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.

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



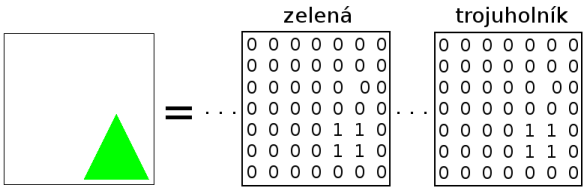
Obr. 3: Tenzorová reprezentácia celej scény.

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.

Táto tenzorová reprezentácia je vytvorená pre každý objekt na scéne a matice, ktoré reprezentujú tú istú črtu, sú sčítané. Detailná ukážka je na obrázku 3, ktorý demonštruje zakódovanie scény pozostávajúcej z červeného kruhu, modrého štvorca a modrého trojuholníka.

Takto sa zakóduje celá scéna, ale ostáva ešte kódovanie označeného objektu. Vtedy stačí len pozícia objektu, nakoľko nie je potrebné jeho črty kódovať dvakrát. Pozícia objektu sa zaznamenáva v matici 2×2, v ktorej sa nachádza práve raz číslo 1, ktoré určuje v ktorom rohu scény sa objekt nachádza. Celá vstupná vrstva teda pozostáva zo 6 matíc reprezentujúcich čŕt objektov a z 1 matice určujúcej pozíciu označeného objektu.

Pretože model navrhnutý Blankom (ktorý je opísaný v ďalšej časti) je obmedzený tým, že vyžaduje aby matica určujúca pozíciu označeného objektu bola dostatočne veľká, pri experimentoch nepoužívame opísanú tenzorovú reprezentáciu, ale *zväčšenú* tenzorovú reprezentáciu. Matica 2×2 je nahradená maticou 7×7, kde namiesto jednej jednotky sú použité 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 obrázku 4, kde je použitá zväčšená tenzorová



Obr. 4: Zväčšená tenzorová reprezentácia zeleného trojuholníka.

reprezentácia zeleného trojuholníka umiestneného na scéne vpravo dole. Matice, ktoré nie sú podstatné (a obsahujú iba samé nuly) sú vynechané. Výsledná reprezentácia pre všetky objekty na scéne vznikne rovnako – sčítaním týchto zväčšených tenzorových reprezentácií každého objektu.

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). Jeden z bodov kritiky explicitnej reprezentácie sme uviedli v časti 2.2.1. 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 1, ktorú možno explicitne reprezentovať[3] ako:

```
zdrojová-scéna:
  unikátny-objekt(červ. troj.)
  ostatné-objekty(červ. kruh, červ. kruh)
cieľová scéna:
  unikátny-objekt(mod. štv.)
  ostatné-objekty(mod. kruh, zel. 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.

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

Obr. 5: Explicitná reprezentácia zeleného štvorca.

Problém nastáva vtedy, ak by bola použitá táto reprezentácia (ktorá je ekvivalentná s tou predchádzajúcou):

```
zdrojová-scéna:
  unikátny-objekt(červ. troj.)
  ostatné-objekty(červ. kruh, červ. kruh)
cieľová scéna:
  unikátny-objekt(zel. kruh)
  ostatné-objekty(mod. kruh, modrý štv.)
```

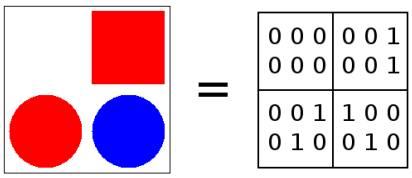
Je vidieť, že pri tejto reprezentácií by nebola vytvorená správna analógia.

Naša explicitná reprezentácia začína nad úrovňou problému viazania (angl. binding problem)[4], 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 nie je ťažké prerobiť implicitnú reprezentáciu, ktorá je považovaná za jedinú korektnú reprezentáciu pri analógiách, získanú percepciou 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 5 znázorňuje toto kódovanie.

Poradie týchto šestíc určuje pozíciu objektu. Prvá šestica reprezentuje ľavý horný objekt, druhá šestica pravý horný objekt, tretia šestica ľavý dolný objekt a nakoniec štvrtá šestica reprezentuje pravý dolný objekt. Ak sa objekt na nejakom mieste nenachádza, je použitých šesť núl. Príklad celej scény reprezento-

[4]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)



Obr. 6: Explicitná reprezentácia celej scény.

vanou týmto spôsobom je na obrázku 6, kde vektor šestíc je kvôli prehľadnosti rozdelený na dva riadky.

Pozícia označeného objektu sa definuje rovnako ako pri tenzorovej reprezentácií (maticou 2×2, alebo 7×7). Takisto je možné použiť zväčšenú reprezentáciu, ktorá pozostáva zo 49 bitov na určenie pozície (reprezentácia scény sa nemení – zostáva 6×4 = 24 bitov).

3 Model Analogator

Konekcionistický model *Analogator* bol navrhovaný tak, aby sa dokázal učiť sa analógie typu *časť–celok* 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.

Blank 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 používa 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 7.

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.

Váhy siete sú v procese trénovania siete nastavované algoritmom spätného šírenia chyby (angl. backpropagation algorithm), 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

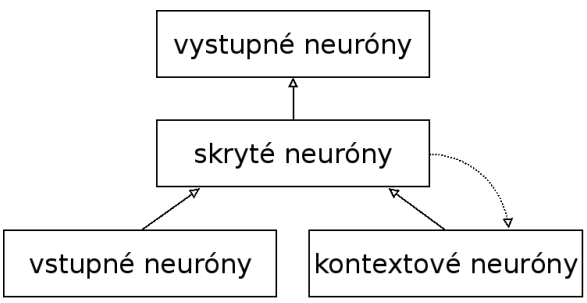
Následne sa váhy modifikujú podľa týchto pravidiel (hodnota α je rýchlosť učenia a μ je moment):

- $\Delta w_{ik}(t+1) = \alpha\delta_i h_k + \mu\Delta w_{ik}(t)$ 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 Architektúra základného modelu

Model *Analogator* má rovnakú architektúru ako jednoduchá rekurentná sieť opísaná v predchádzajúcej časti (obrázok 7). 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 časovom kroku), alebo kópia skrytej vrstvy (v druhom časovom kroku). Výstupná vrstva obsahuje pozíciu označeného objektu a pozície ostatných, neoznačených objektov v celej scéne. Výpočet pre vytvorenie analógie pozostáva z dvoch krokov:

1. Do prvej časti vstupnej vrstvy je uložená reprezentácia scény (črty objektov a ich pozície) pozostávajúcej z viacerých objektov a do druhej časti (ktorá zároveň zohráva úlohu kontextových neurónov) je uložená pozícia objektu v scéne,



Obr. 7: Architektúra jednoduchej rekurentnej siete.

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 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 ostatných objektov v druhej scéne.

Obmedzenie, ktoré vyplýva z tejto architektúry je to, 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 stačiť preto, aby sa sieť dokázala učiť. Vyšší počet neurónov na skrytej vrstve zase núti pridávať dodatočné a nepotrebné neuróny na vstupnú vrstvu.

3.1.1 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. 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 pre to výstup a následne sa váhy upravia, aby bola scéna správne rozdelená na označený objekt a neoznačené objekty. 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, alebo ak sa dosiahla nulová trénovacia chyba. 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.

3.2 Modifikácie základného modelu

Náš prínos spočíval v tom, že sme odstránili obmedzenie základného modelu, ktoré nútilo, aby veľkosť skrytej vrstvy a časti vstupnej vrstvy kódujúcej pozíciu označeného objektu bola totožná. Vyriešili sme to dvomi spôsobmi:

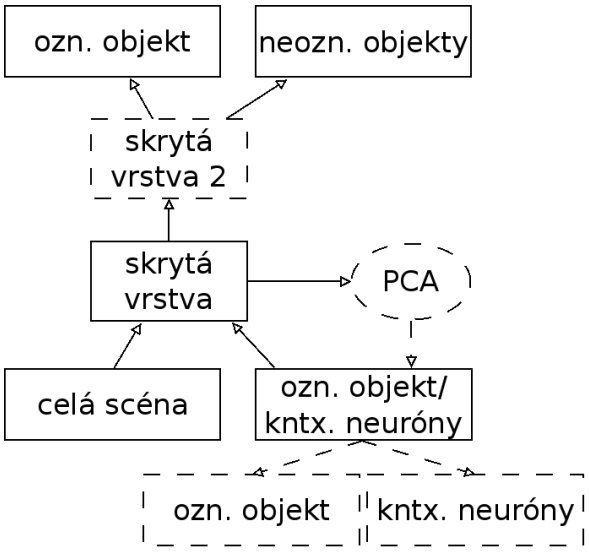
1. Oddelenie neurónov, ktoré zohrávajú dve rôzne úlohy na vstupnej vrstve – neuróny pre reprezentáciu pozície označeného objektu a kontextové neuróny. Potom vo výpočte sú v prvom kroku kontextové neuróny nastavené na nulové hodnoty (v druhom kroku obsahujú kópiu skrytej vrstvy z prvého kroku) a v druhom kroku sú zase nulové neuróny tie, ktoré určovali pozíciu označeného objektu v prvom kroku.

2. Lineárna transformácia skrytej vrstvy pomocou analýzy hlavných komponentov (angl. Principal Component Analysis, PCA) a uloženie výsledku do vstupnej vrstvy. Použili sme algoritmus *GHA* (Sanger, 1989), ktorý upravuje váhy matice W tak, že pri transformácií $\forall i \in \{1,2,...,k\} : y_i = \sum_{j=\{1,2,...,n\}} w_{ij}x_j$ sa stratí čo najmenej informácie (n je veľkosť vstupu – v našom prípade veľkosť skrytej vrstvy – a k je veľkosť výstupu – v našom prípade časť vstupnej vrstvy kódujúcej pozíciu označeného objektu). Celý model, s touto modifikáciou, sa učí súčasne algoritmom spätného šírenia chyby a algoritmom *GHA*.

Ďalšia modifikácia bola pridanie skrytej vrstvy medzi existujúcu skrytú vrstvu a výstupnú vrstvu. To umožní modelu lepšie sa učiť analógie, rýchlejšie konvergovať a viac generalizovať. Kombináciou týchto modifikácií sme spolu získali týchto 5 výsledných modelov[5]:

1. model s *PCA*

[5]Nemá zmysel kombinovať *PCA* s oddelenou vstupnou vrstvou, nakoľko oba riešia ten istý problém.



Obr. 8: Pridané modifikácie základného modelu. Celou čiarou sú znázornené elementy nachádzajúce sa u všetkých modelov (aj u základného). Prerušovanou čiarou sú elementy o ktoré sme základný model rozšírili.

2. model s oddelenou vstupnou vrstvou
3. model s ďalšou skrytou vrstvou
4. model s *PCA* a ďalšou skrytou vrstvou
5. model s oddelenou vstupnou vrstvou a ďalšou skrytou vrstvou

Všetky modifikácie sú zhrnuté na obrázku 8. V tomto príspevku sme si vybrali model s *PCA* a model s oddelenou vstupnou vrstvou a ďalšou skrytou vrstvou, ktorých správanie sme vyhodnotili na dvoch typoch analógie.

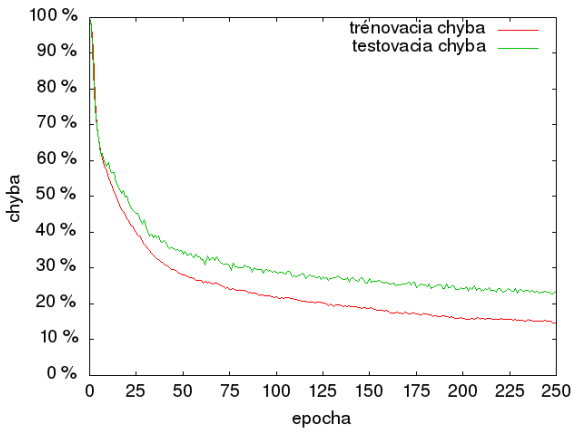
4 Experimenty

Pretože naučiť sieť jeden typ analógie je ľahká úloha (demonštroval to Blank vo svojej dizertačnej práci), v tejto práci sme sa tomu nevenovali. Namiesto toho bolo našim cieľom, aby sa nejaký nami modifikovaný model dokázať naučiť oba typy analógie, ktoré boli vysvetlené v časti 2.1.1, pretože majú ku sebe veľmi blízko a človek medzi nimi v zásade nevidí rozdiel. Pred tým než vyhodnotíme správanie nových modelov, pozrieme sa na správanie základného modelu s Blankovou implicitnou reprezentáciou. Aby sme zachytili približné všeobecné správanie daného modelu, model bol učený 10 krát a výsledný graf s trénovacou a testovacou krivkou bol priemerom 10 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 sa najprv vygenerovali všetky vzory a z tých vzorov bolo náhodne vybratých 10000 vzorov, z ktorých 90% bolo použitých na trénovanie a 10% na testovanie. Parametre algoritmu spätného šírenia chyby boli: rýchlosť učenia $\alpha = 0.1$ a moment $\mu = 0.7$.

4.1 Základný model s implicitnou reprezentáciou

Použitá reprezentácia vstupov bola zväčšená tenzorová, čo znamená, že na vstupnej vrstve bolo 49×6 = 294 neurónov reprezentujúcich celú scénu, 49 neurónov pre určenie pozície označeného objektu a 1 *bias* neurón. Spolu 344 neurónov na vstupnej vrstve. Skrytá vrstva mala veľkosť 49+1 = 50 neurónov a výstupná vrstva mala 49+49 = 98 neurónov.

Model sa aj napriek veľkému množstvu epoch (v porovnaní s ďalšími modelmi) nedokázal naučiť oba typy analógií. Nižšie (obrázok 9) je možné vidieť správanie modelu.



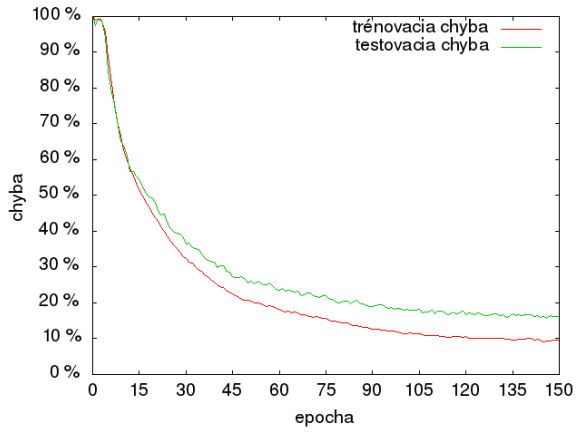
Obr. 9: Výsledok učenia základného modelu s implicitnou reprezentácii na dvoch typoch analógie.

4.2 Základný model s explicitnou reprezentáciou

Tento experiment mal ukázať akú zmenu prinesie nami navrhnutá reprezentácia problému geometrických analógií. Na vstupe bolo 24 neurónov po-

pisujúcich kompletnú scénu a 49 neurónov, ktoré uchovávali informáciu o pozícii označeného objektu. Vstupná vrstva mala teda veľkosť 24+49+1=74. Skrytá a výstupná vrstva boli rovnako veľké ako pri základnom modeli s implicitnou reprezentáciou, čiže 50 neurónov na skrytej vrstve a 98 neurónov na výstupnej vrstve.

Obrázok 10 zobrazuje podstatné zlepšenie, aj keď stále sa nedá povedať, že model sa dokázal naučiť oba typy analógií. Z grafu môžeme usúdiť, že explicitná reprezentácia má pri základnom modeli pozitívny prínos. Testovacia chyba je nižšia približne o 5%. Model zároveň potrebuje podstatne menší počet epoch, aby minimalizoval chybu, a potrebuje menej času na jednu iteráciu (kvôli menšiemu vstupu) oproti základnému modelu pracujúceho s implicitnou reprezentáciou.

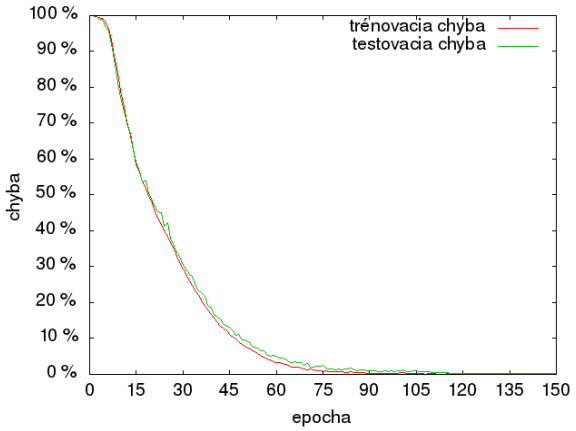


Obr. 10: Výsledok učenia základného modelu s explicitnou reprezentácii na dvoch typoch analógie.

4.3 Model s oddelenou vstupnou vrstvou a ďalšou skrytou vrstvou s explicitnou reprezentáciou.

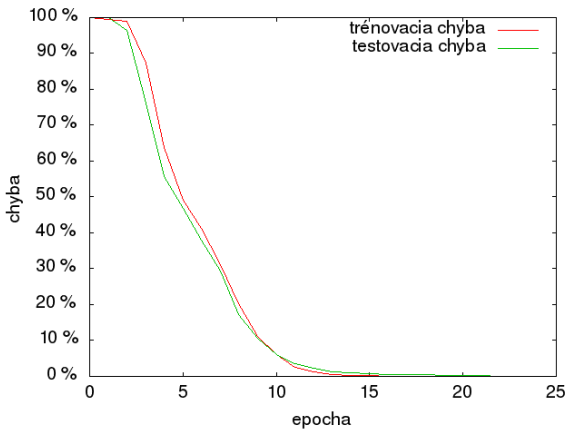
Tento model pracuje lepšie s väčšou trénovacou množinou ako predchádzajúce modely, preto sme použili trénovaciu množinu o veľkosti 17100. Testovaciu množina sme adekvátne zvýšili na veľkosť 1900. Použili sme 40 neurónov na prvej skrytej vrstve (takisto kontextová časť vstupnej vrstvy mala 40 neurónov) a 50 neurónov na druhej skrytej vrstve.

Simulácie ukázali, že tento model sa už vie úspešne naučiť oba typy analógií, čo je obrovské zlepšenie. Správanie síce nie je perfektné (testovacia chyba po 150. epoche nebola 0%, ale 0.1%), no veríme, že testovacia chyba sa dá znížiť ešte viac správnym zvolením parametrov modelu (nám sa také parametre nepodarilo nájsť). Priebeh simulácií je znázornený na obrázku 11.



Obr. 11: Výsledok učenia modelu s oddelenou vstupnou vrstvou a ďalšou skrytou vrstvou s explicitnou reprezentáciou na dvoch typoch analógie.

4.4 Model s *PCA* s implicitnou reprezentáciou



Obr. 12: Výsledok učenia modelu s *PCA* s implicitnou reprezentáciou na dvoch typoch analógie.

V modeli sme použili 300 neurónov na skrytej vrstve, ktoré boli lineárne transformované do 49 neurónov na vstupnej vrstve. Rýchlosť učenia *PCA* sme nastavili na hodnotu $\alpha = 0.0001$. Vyššie hodnoty 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

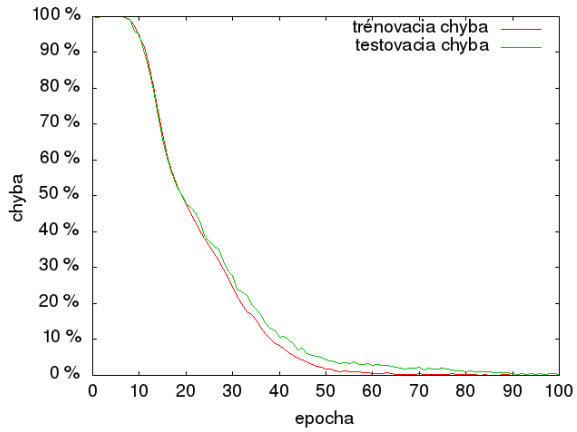
100%.

Oproti predchádzajúcemu modelu je tento model približne šesťkrát lepší v počte epoch. Trénovacia chyba v poslednej epocha bola 0.05%, čo je v priemere pol vzora na celú testovaciu množinu. Správanie je zobrazené na obrázku 12.

4.5 Model s *PCA* s explicitnou reprezentáciou

Tento experiment mal ukázať, či explicitná reprezentácia má prínos aj pri modeli s *PCA*. Parametre sú identické ako pri modeli s implicitnou reprezentáciou: 300 neurónov na skrytej vrstve transformovaných do 49 neurónov na vstupnej vrstve a rýchlosť učenia $\alpha = 0.0001$.

Model má zjavne horší výkon (obrázok 13) ako predchádzajúci model. Testovacia chyba už bola o čosi vyššia, 0.15%, a počet epoch narástol približne štvornásobne.



Obr. 13: Priemer 10 simulácií modelu s *PCA* s explicitnou reprezentáciou na dvoch typoch analógie.

5 Záver

Nami navrhnuté modifikácie majú veľký prínos v modeli *Analogator*. Model, ktorý ich používa, prestáva byť závislý na jednom type analógie, ale je možné ho učiť aj na dvoch typoch, čo je krok vpred v budovaní analogického uvažovania strojov. Najlepší model, ktorý sme našli, je model rozšírený o *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. Otázka, ktorá zostala nezodpovedaná je tá, či by model dokázal fungovať aj pri troch, či viacerých typoch analógie. 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.

Literatúra

Blank, D. S. (1997). *Learning to See Analogies: A Connectionist Exploration*.

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.

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

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, USA.

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.

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, USA.

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