Konekcionistický prístup k porozumeniu textu na báze situačného priestoru mikrosveta

Diplomová práca

Peter Jankovič **2008**

Konekcionistický prístup k porozumeniu textu na báze situačného priestoru mikrosveta

DIPLOMOVÁ PRÁCA

Peter Jankovič

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
KATEDRA APLIKOVANEJ INFORMATIKY

Kognitívna veda

Školiteľ diplomovej práce:
doc. Ing. Igor Farkaš, PhD.

BRATISLAVA, 2008

Čestne vyhlasujem, že som diplomovú prácu vypracoval samostatne, len s použitím uvedenej literatúry a pod odborným dohľadom školiteľa.

Bratislava, jún 2008

….....

Peter Jankovič

Ďakujem svojmu školiteľovi Igorovi Farkašovi za cenné rady a pripomienky, ktoré mi poskytol pri písaní diplomovej práce. Zároveň sa chcem poďakovať rodine a priateľom, že mi boli oporou pri písaní práce, ako aj počas celého štúdia.

Abstrakt

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY
KATEDRA APLIKOVANEJ INFORMATIKY

Autor:	Peter Jankovič
Názov diplomovej práce:	Konekcionistický prístup k porozumeniu textu na báze situačného priestoru mikrosveta
Školiteľ diplomovej práce	doc. Ing. Igor Farkaš, PhD.
Rozsah práce:	57 strán

Bratislava, jún 2008

Diplomová práca sa venuje konekcionistickému modelovaniu porozumenia textu na situačnej úrovni reprezentácie, v rámci uvažovaného, dobre opísaného jednoduchého mikrosveta. Porozumenie textu chápeme ako priradenie (pomocou rekurentnej neurónovej siete) distribuovaných situačných reprezentácií vstupných vetám, opisujúcim situácie v mikrosvete. Teoretický model je implementovaný v programovacom jazyku JAVA, čo nám umožnilo skúmať reprezentačné vlastnosti distribuovaného situačného priestoru a testovanie generalizačnej vlastnosti modelu neurónovej siete.

Kľúčové slová: porozumenie textu, mikrosvet, model distribuovaného situačného priestoru, samoorganizujúca sa mapa, jednoduchá rekurentná sieť

Predhovor

V diplomovej práci sa venujeme porozumeniu textu, ktorý opisuje situácie z tzv. mikrosveta (zjednodušený, systematicky opísateľný svet, s presne definovanými pravidlami). V procese porozumenia vytvárame tzv. situačnú reprezentáciu textu (subsymbolový prístup), narozdiel od tradične budovanej propozičnej (symbolovej) reprezentácie (napr. [1]). Najskôr musíme mať nástroj na vytvorenie situačných reprezentácií, ktoré zodpovedajú situáciám v mikrosvete. Distribuovaný situačný priestor získavame pomocou samoorganizujúcej sa mapy natrénovanej na príkladoch situácií z mikrosveta. Následne trénujeme rekurentnú neurónovú sieť, aby dokázala priradiť vetám odpovedajúcu situačnú reprezentáciu, čo interpretujeme ako porozumenie textu. Tu je kľúčovou požadovanou vlastnosťou schopnosť zovšeobecnenia rekurentnej siete, t.j. porozumenie aj nových situácií ako prejav systematickosti reprezentácií v situačnom priestore.

Inšpiráciu pri tvorbe práce som čerpal z [2] a [3]. Najskôr som si naštudoval relevantné zdroje, aby som získal prehľad o problematike. Potom som navrhol model systému a následne ho aj naprogramoval. Implementácia predstavuje podstatnú časť tejto práce, pretože na jej základe overujeme funkčnosť nášho modelu. Výsledky dosiahnuté pomocou softvérovej aplikácie prezentujeme v tejto práci.

V prehľadovej kapitole uvádzame hlavné poznatky a nástroje využité aj v našej práci. V nasledujúcej kapitole definujeme náš mikrosvet. V ďalšej kapitole podrobne opisujeme situačnú reprezentáciu a skúmame jej vlastnosti. V poslednej kapitole sa venujeme problému porozumenia jazyka.

Pri teoretickej príprave modelu, ako aj pri písaní konečnej podoby práce mi veľmi pomohol môj školiteľ Igor Farkaš. Svojimi cennými radami, pripomienkami a motivovaním výraznou mierou prispel k skvalitneniu tejto práce.

Obsah

Abstrakt.....5

Predhovor.....6

1 Úvod.....9

2 Prehľad.....11

2.1 Situačná reprezentácia.....11
2.2 Znalosti o svete v počítačových modeloch porozumenia textu.....13
2.3 Distribuovaný situačný priestor 15
2.3.1 Mikrosvet.....15
2.3.2 Situačná reprezentácia.....16
2.4 Umelé neurónové siete.....17
2.4.1 Samoorganizujúca sa mapa.....18
2.4.2 Perceptrón.....19
2.4.3 Rekurentná neurónová sieť.....19

3 Mikrosvet.....20

3.1 Kódovanie mikrosveta.....22

4 Situačná reprezentácia.....24

4.1 Samoorganizujúca sa mapa.....24
4.2 Vizualizácia.....28
4.2.1 Distribuovaná reprezentácia.....29
4.3 Klasifikácia situácií z mikrosveta.....34
4.4 Škálovanie mikrosveta.....36

5 Jazyk.....39

5.1 Generátor jazyka.....39
5.2 Porozumenie jazyka.....41

5.2.1 Trénovacia stratégia 1....41
5.2.2 Trénovacia stratégia 2....43

6 Záver....45

Príloha A – Implementácia....47

BlockWorld....47
JavaSOM....48
WeightsMap....49
BlockWorldVisualisation....49
nnsim....51
BlockWorldVerbalization....53

Príloha B – Obsah CD....55

Referencie....56

1 Úvod

V oblasti porozumenia textu je stále otvorenou otázkou, aký typ mentálnej reprezentácie si čitateľ vytvára pri tomto procese. Popri *klasickej* propozičnej úrovni získavajú čoraz viac na význame tzv. situačné modely, podporované aj empirickými výskumami ([1]). Porozumenie textu nemôžeme chápať iba ako vytváranie reprezentácie textu samotného. Dôležité je tiež skonštruovanie modelu situácie opísanej textom.

Dôvodov pre prijatie situačných modelov je hneď niekoľko. Svoje opodstatnenie majú pri porozumení dlhších textov – príbehov. Slúžia na integrovanie informácií z jednotlivých viet. Dokážeme pomocou nich vysvetliť podobnosti pri vnímaní situácie rôznymi zmyslami, pričom vytvorená situačná reprezentácia je veľmi podobná. Automatický prekladač, ktorý uplatňuje situačný model textu môže dosiahnuť lepšie výsledky ako tradičný systém, ktorý prekladá individuálne slová príp. vety bez reprezentovania situácie vykreslenej v texte.

V tejto práci sa zaoberáme vytvorením situačnej reprezentácie jednoduchého prostredia – tzv. mikrosveta. Boli sme inšpirovaní prácami [2] a [3]. Situačnú reprezentáciu vytvárame podobným spôsobom, opisujeme však iný mikrosvet. Náš mikrosvet obsahuje tri rôzne objekty rozmiestnené na mriežke veľkosti 3×3, pričom poloha objektov nie je ľubovoľná, ale spĺňa vopred definované pravidlá. Situačná reprezentácia vzniká na základe rozmiestnenia objektov na scéne a nie verbálneho opisu scény. Nie je teda závislá od propozičnej štruktúry textu, čo je jeden zo základných predpokladov dobrého situačného modelu.

Výhodou mikrosveta oproti reálnemu prostrediu je, že ho máme plne pod kontrolou a môžeme implementovať všetky znalosti, ktoré v ňom platia. Skúmame a analyzujeme rôzne aspekty reprezentácie. Vzniknutú situačnú reprezentáciu použijeme v procese porozumenia jazyka. Keďže máme zachytené všetky zákonitosti platné v mikrosvete, môžeme testovať schopnosť porozumenia na všetkých textoch opisujúcich náš mikrosvet.

V práci sa zaoberáme len vlastnosťami situačnej reprezentácie v jednom časovom okamihu. Práca by sa dala rozšíriť o zohľadnenie zmien situácií v čase. Najskôr by bolo treba nájsť vhodnú reprezentáciu časových závislostí. Model by mohol byť následne použitý na porozumenie textu opisujúceho postupnosť situácií. Ďalšiu pozornosť by si tiež zaslúžilo skúmanie možnosti škálovania mikrosveta na svet väčších rozmerov.

2 Prehľad

V tejto kapitole uvádzame prehľad predchádzajúcich prác a výsledkov, o ktoré sa pri písaní práce opierame. Najprv zhrnieme vlastnosti situačných modelov a dôvody, prečo je vhodné sa nimi zaoberať pri porozumení textu. Odôvodníme tiež, prečo sme si zvolili stratégiu opisu mikrosveta namiesto reálneho prostredia. Bližšie opíšeme konkrétny situačný model – distribuovaný situačný priestor. Nakoniec stručne charakterizujeme dva použité typy neurónových sietí – samoorganizujúcu sa mapu a perceptrón.

2.1 Situačná reprezentácia

V teórii porozumenia textu rozlišujeme vo všeobecnosti tri typy reprezentácií vytvárané pri tomto procese. Na najnižšej úrovni je povrchová reprezentácia (*surface representation*), ktorá pozostáva len zo slov textu. Druhý stupeň je propozičná reprezentácia (*textbase* alebo *propositional representation*). Text je na tejto úrovni reprezentovaný sieťou propozícií zodpovedajúcich jednotlivým slovám. Propozície sú v sieti prepojené, ak zdieľajú nejaký argument (slovo alebo viac slov). Na najvyššej úrovni v hierarchii je situačná reprezentácia (*situational representation* resp. *situation model*), opisujúca celkový stav vecí – situáciu.

Samotná povrchová reprezentácia pre porozumenie textu zrejme nestačí. Otvorenou otázkou ale stále ostáva, či má mentálna reprezentácia textu propozičný alebo situačný charakter. Až do skorých 80-tych rokov minulého storočia bolo

porozumenie textu spájané s vytváraním reprezentácie textu samotného (teda propozičnej reprezentácie), a nie situácie opísanej v texte. V roku 1983 upozornili na nutnosť reprezentácie situácií nezávisle od seba Johnson-Laird (mentálne modely, [4]) a Dijk a Kintsch (situačné modely, [5]). V obidvoch modeloch je ale popri reprezentácii situácií zdôrazňovaná aj potreba reprezentácie textu samotného.

Dôvody, prečo je situačná reprezentácia nevyhnutná pri porozumení textu, sú zhrnuté v [1]. Niektoré z nich tu spomenieme a bližšie opíšeme:

- integrácia informácií z viacerých viet
- vnímanie situácií rôznymi zmyslami
- vplyv znalosti domény na schopnosť porozumenia
- vysvetlenie prekladu
- získavanie informácií z viacerých dokumentov

Ak máme porozumieť dlhšiemu príbehu, potrebujeme pospájať informácie z jednotlivých viet. Tá istá osoba alebo vec môže byť v príbehu označená odlišnými výrazmi. V texte tiež nemusia byť spomenuté všetky relevantné informácie. Častokrát musíme doplniť niektoré údaje potrebné k pochopeniu príbehu z dlhodobej pamäte alebo inak zistiť na základe dostupných dát (napr. pomocou inferencie).

Vytváranie situačných modelov pri porozumení textu je podporené aj empirickými výskumami. Ak vety v texte na seba nadväzujú, ale neopisujú unikátnu situáciu, porozumenie textu je značne sťažené. Čitateľ text nepochopí práve preto, že sa mu nepodarí vytvoriť jednoznačný situačný model textu.

Situačná reprezentácia sa nevzťahuje iba na porozumenie písaného textu, ale dajú sa pomocou nej vysvetliť aj pozorované podobné výsledky pri vnímaní situácie rôznymi zmyslami (zrak, sluch). Experimenty dokazujú, že pri vnímaní tej istej situácie odlišnými spôsobmi (konkrétne hovorený príbeh a krátky film) sa u subjektov vytvorí veľmi podobná situačná reprezentácia. Porozumenie je teda nezávislé od formy, akou sme so situáciou oboznámení. Zrejme existuje súbor kognitívnych procedúr slúžiacich na vytváranie situačných modelov, ktoré sú nezávislé od druhu zmyslového vnímania.

Niekedy nevieme vysvetliť rozdiely v schopnosti porozumenia len na základe verbálnych dispozícií. Verbálne menej zdatné osoby dokážu predčiť schopnejších jedincov vďaka väčším skúsenostiam s danou problematikou. Tento jav bol experimentálne zachytený napríklad pri porovnávaní schopnosti porozumenia textu o futbalovom zápase ([6]). Testované osoby boli rozdelené do viacerých skupín podľa ich

futbalových znalostí. Znalci futbalu si zapamätali viac informácií z textu ako nováčikovia. Toto pozorovanie nemôže byť objasnené len na propozičnej úrovni reprezentácie, pretože predkladaný text bol pre všetkých rovnaký. Vo všeobecnosti si doménoví experti dokážu jednoduchšie vybudovať situačný model textu ako ľudia neznalý danej domény.

Preklad je ďalšia oblasť, kde možno cítiť potrebu situačných modelov. Dôkazom je aj množstvo neúspešných automatických prekladačov prirodzených jazykov. Tieto systémy v drvivej väčšine pracovali len na propozičnej úrovni reprezentácie textu. Prirodzený jazyk má však veľa zákutí, kde je preklad od slova do slova nežiadúci. Jedným z problematických prvkov jazyka pri tomto prístupe sú príslovia. Tie sa nedajú preložiť priamo, obyčajne majú v každom jazyku inú podobu, ale rovnaký (podobný) význam. Pri preklade treba vytvoriť model situácie opísanej v pôvodnom jazyku a následne túto situáciu vyjadriť v cieľovom jazyku.

Informácie obvykle nezískavame len z jediného zdroja, ale zdroje musíme kombinovať a informácie v nich obsiahnuté integrovať. Situačný model teda nevzniká naraz, ale postupne. Vytváranie situačného modelu vyžaduje často aj revidovanie už prijatých údajov, aby spolu tvorili koherentný celok. K informáciám sa treba tiež stavať kriticky, pretože autor dokumentu mohol kvôli predsudkom alebo iným dôvodom fakty pozmeniť.

Tieto dôvody pre prijatie situačnej reprezentácie textu pokladáme za dostatočne silné, a preto sa v tejto práci venujeme skúmaniu vlastností situačnej reprezentácie. Bolo by samozrejme veľmi zložité reprezentovať text v prirodzenom jazyku a všetky komplexné znalosti nutné na jeho pochopenie. V nasledujúcej kapitole popíšeme stratégiu, ako sa s týmto problémom vysporiadať.

2.2 Znalosti o svete v počítačových modeloch porozumenia textu

V [7] sú preštudované štyri prístupy, ako možno implementovať bázu znalostí v počítačových modeloch vyšších kognitívnych procesov, medzi ktoré istotne spadá aj porozumenie textu. Sú to tieto štyri stratégie:

- žiadne znalosti o svete

- ad hoc výber znalostí
- výber znalostí z textového korpusu
- mikrosvet

Výsledky štúdie možno zhrnúť nasledovne: zdanlivo úspešné výsledky modelu sú nedostatočné ak nie sú znalosti implementované, prípadne sú vyberané ad hoc. Znalosti extrahované z textového korpusu nie sú vhodné pre modely porozumenia reči. Vhodným riešením sa javí byť práve stratégia mikrosveta.

Na tomto mieste iba stručne uvedieme dôvody predchádzajúcich tvrdení. Každý model porozumenia textu by mal byť schopný spracovať všetky relevantné znalosti o svete. V praxi sa často stáva, že model sa zrúti ak sa pridajú relevantné znalosti, na spracovanie ktorých nebol navrhnutý.

Ak obmedzíme znalosti o svete iba na informácie, ktoré sa priamo vyskytli v konkrétnom texte, ťažko môžeme otestovať všeobecné vlastnosti systému, prípadne overiť jeho správanie na experimentálnych dátach.

Hoci automatickou extrakciou dát z textového korpusu získame znalosti, ktoré sú vo všeobecnosti aplikovateľné, nie sú vhodné v prípade porozumenia dlhšieho textu (príbehu). Tieto znalosti totiž zachytávajú významy slov, ale nie kauzálne vzťahy vo svete, ktoré sú pre pochopenie príbehu nevyhnutné.

Keďže implementácia dostatočného množstva znalostí o reálnom svete je len ťažko predstaviteľná, volíme stratégiu opisu všetkých znalostí zjednodušeného sveta – mikrosveta. Znalosti získavame opäť z korpusu, tentokrát však nejde o textový korpus, ale o *korpus situácií mikrosveta*.

Modely založené na znalostiach mikrosveta môžu byť testované na rôznych textoch (o situáciách v mikrosvete) bez nutnosti nastavovania parametrov modelu. Výsledky sú teda nezávislé od konkrétneho použitého textu, na ktorý bol model ideálne nastavený.

Možným nedostatkom stratégie mikrosveta je jeho ťažšia rozšíriteľnosť. Ak však chceme stratégiu mikrosveta striktne dodržať, musíme všetky znalosti implementovať, čo môže byť vo svete väčších rozmerov problematické. Na druhej strane nás to ale chráni pred vágnym definovaním sveta.

Ďalším problémom je, že model nedokáže spracovať texty o situáciách mimo mikrosveta. Napriek tomu ale môže poskytnúť zaujímavé výsledky v oblasti porozumenia textu.

2.3 Distribuovaný situačný priestor

Hlavnou inšpiráciou tejto práce bol distribuovaný situačný priestor (DSP) opísaný v článku [2]. V našej práci je situačná reprezentácia veľmi podobná, opisujeme ale odlišný mikrosvet. Na tomto mieste stručne opíšem pôvodný mikrosvet a jeho situačnú reprezentáciu vo forme DSP.

2.3.1 Mikrosvet

V mikrosvete existujú dve postavy – Bob a Jilly. Situácia je definovaná ich aktivitami, stavmi, príp. stavmi mikrosveta. Na opis situácie slúži 14 základných propozícií: `Sun` (slnko svieti), `Rain` (prší), B outside (Bob je vonku), `J outside` (Jilly je vonku), `Soccer` (Bob a Jilly hrajú futbal), `Hide-and-seek` (Bob a Jilly sa hrajú na schovávačku), `B computer` (Bob hrá počítačovú hru), `J computer` (Jilly hrá počítačovú hru), `B dog` (Bob sa hrá so psom), `J dog` (Jilly sa hrá so psom), `B tired` (Bob je unavený), `J tired` (Jilly je unavená), `B wins` (Bob vyhráva), `J wins` (Jilly vyhráva).

Vo svete platí niekoľko obmedzení. Bob a Jilly môžu hrať futbal len keď sú vonku a počítačovú hru len vo vnútri. Naraz môžu vykonávať len jednu aktivitu. Niektorý z nich môže vyhrať len ak hrajú futbal, schovávačku, alebo obaja hrajú počítačovú hru. Okrem týchto pevných pravidiel sú vo svete aj jemnejšie obmedzenia. Je pravdepodobnejšie, že Bob a Jilly budú na tom istom mieste vykonávať rovnakú aktivitu, ako to, že sú na rôznych miestach a vykonávajú rôzne činnosti. Všetky tieto obmedzenia definujú znalosti o svete, ktoré sú nezávislé od časovej dimenzie.

Okrem toho sú tu ešte časové znalosti. Pevné obmedzenia sú, že Bob a Jilly prestanú hrať hru akonáhle jeden z nich vyhrá a vyhrať sa dá len ak hrali hru aj v predchádzajúcom časovom kroku. Dôležitým jemným obmedzením je, že keď je niekto unavený, je jeho výhra menej pravdepodobná.

Na základe týchto obmedzení je vygenerovaných 250 za sebou idúcich situácií. Každá situácia je v tejto fáze reprezentovaná 14-rozmerným vektorom, každý rozmer zodpovedá práve jednej propozícii a definuje, či propozícia v danej situácii platí alebo nie. Na základe tejto sekvencie elementárnych situácií vieme vytvoriť situačnú reprezentáciu, ktorá odráža pravdepodobnosti výskytov situácií v mikrosvete.

2.3.2 Situačná reprezentácia

V tejto časti opíšeme situáciu ako distribuovaný vektor, z ktorého vieme aproximovať subjektívnu pravdepodobnosť platnosti propozície v danej situácii. Táto aproximácia sa nazýva presvedčenie (*belief value*), pretože označuje, do akej miery môžeme byť presvedčení o platnosti propozície v danej situácii.

Na situačnú reprezentáciu sa nemusíme pozerať len ako na vektor, môžeme ju vnímať aj ako dvojrozmernú plochu. Platí pritom, že čím špecifickejšiu situáciu plocha charakterizuje, tým je príslušná plocha menšia. Plocha musí tiež reflektovať logické vzťahy platné medzi propozíciami. Ak máme napríklad reprezentáciu konjunkcie dvoch propozícií, jej plocha by mala byť približne prienik plôch zodpovedajúcich reprezentáciam samostatných propozícií. Rovnako musia ostať zachované aj vzťahy disjunkcie a negácie.

Pre akýkoľvek reálny počet propozícií a situácií nedokážeme ručne vytvoriť dvojrozmernú situačnú reprezentáciu. Máme našťastie k dispozícii samoorganizujúcu sa mapu (SOM, pozri [8]), stručne opísanú ďalej v prehľade.

Pre každú propozíciu *p* a každý neurón *i* je definovaná hodnota príslušnosti (*membership value*) $\mu_i(p)$ nadobúdajúca hodnoty medzi 0 a 1. Jej hodnota označuje, do akej miery daný neurón prispieva k ploche reprezentujúcej danú propozíciu. Počas trénovania sa tieto hodnoty nastavujú, aby odzrkadlovali závislosti medzi propozíciami z trénovacej množiny (250 príkladov situácií).

SOM samozrejme nemusí byť dvojrozmerná, ale je to lepšie pre vizualizáciu. Takisto rozmery mapy (v tomto prípade 10×15 hexagonálnych buniek) nemajú zásadný vplyv na kvalitu reprezentácie. 150 buniek (*n*) tvorí dvojrozmernú mriežku, ale môžeme sa na ne dívať aj ako na dimenzie n-rozmerného stavového priestoru $<0;1>^n$. Plocha mapy pre propozíciu *p* zodpovedá vektoru $\mu(p)=(\mu_1(p),\mu_2(p),\ldots,\mu_n(p))$. Plochu SOM a vektorovú reprezentáciu môžeme ľubovoľne zamieňať podľa toho, čo nám v danom momente viac vyhovuje.

Keďže plocha v SOM zodpovedajúca propozícii je neostro definovaná, musíme sa uchýliť k teórii fuzzy množín, aby sme vypočítali hodnoty negácií, konjunkcií a ďalších komplexných propozícií. Použité sú nasledovné vzťahy:

$$\mu_i(\neg p)=1-\mu_i(p)$$

$$\mu_i(p\wedge q)=\mu_i(p)\mu_i(q)$$

Ďalšie logické spojky sa dajú ľahko vyjadriť aplikovaním negácie a konjunkcie.

Zatiaľ dokážeme ľubovolnú situáciu vyjadriť ako vektor v DSP. Potrebujeme však vedieť postupovať aj opačným smerom – z daného vektora zrekonštruovať situáciu. Vo všeobecnosti to nie je možné, pretože len zopár bodov v DSP presne zodpovedá nejakej kombinácii propozícií. Vieme ale vypočítať pre každú propozíciu presvedčenie, že platí v danej situačnej reprezentácii.

Nech $X=(x_1, x_2, \ldots, x_n)$ je situačný vektor (alebo plocha SOM) reprezentujúca situáciu *X*. Subjektívna nepodmienená pravdepodobnosť, že sa situácia *X* vyskytne v mikrosvete, je rovná časti mapy, ktorú pokrýva. Toto presvedčenie označíme $\tau(X)$ a vypočítame ho:

$$\tau(X)=\frac{1}{n}\sum_i x_i$$

Potrebujeme tiež vedieť vypočítať subjektívnu pravdepodobnosť propozície *p* v situácii *X*. Túto hodnotu označíme $\tau(p|X)$ a nazveme presvedčenie o propozícii *p* v situácii *X*. Vypočíta sa:

$$\tau(p|X)=\frac{\sum_i \mu_i(p)\, x_i}{\sum_i x_i}$$

Okrem znalostí platných v jednom časovom kroku sú v [2] uvažované aj temporálne znalosti. Tomu sa ale v našom prehľade nebudeme venovať, keďže to pre našu prácu nie je podstatné.

2.4 Umelé neurónové siete

Umelé neurónové siete sú typickým nástrojom používaným v subsymbolovej umelej inteligencii. Základnými stavebnými prvkami sú neuróny, ktoré sú poprepájané pomocou synapsií s adaptovateľnými váhami, tvoriac tak umelú neurónovú sieť.

Umelé neurónové siete nájdu svoje uplatnenie pri riešení rôznych typov úloh. Sú vhodným prostriedkom na rozpoznávanie, zapamätanie alebo klasifikáciu vzorov. Používajú sa tiež na zmenu reprezentácie údajov – či už ide o kompresiu alebo transformáciu dát.

Rozlišujeme dva základné trénovacie mechanizmy umelých neurónových sietí – učenie s učiteľom a bez učiteľa. V prvom prípade sieti spolu so vstupom predkladáme aj

požadovaný výstup. V druhom prípade nechávame sieť, nech si sama vytvorí vnútornú reprezentáciu trénovacej množiny – ide o proces samoorganizácie.

V tomto prehľade bližšie spomenieme jeden typ siete trénovanej bez učiteľa (samoorganizujúca sa mapa) a dva typy siete trénovanej s učiteľom (perceptrón a rekurentná neurónová sieť).

2.4.1 Samoorganizujúca sa mapa

Samoorganizujúca sa mapa (SOM) sa nazýva podľa svojho autora tiež Kohonenova sieť ([8]). Je príkladom siete trénovanej bez učiteľa. Sieť teda nemá počas trénovania k dispozícii požadované výstupy, ale váhy sa adaptujú tak, aby odrážali štatistické vlastnosti trénovacej množiny. Dôležitou črtou SOM je, že ju dokážeme natrénovať tak, aby vstupno-výstupné zobrazenie zachovávalo topológiu. To znamená, že pre ľubovolné dva vzory blízke vo vstupnom priestore sú najviac aktívne neuróny, ktoré sú si tiež fyzicky blízke (vo výstupnom priestore).

Algoritmus trénovania SOM popíšem len schematicky ([9]). V prvom kroku, nazývanom súťaženie, sa hľadá neurón, ktorý najviac reaguje na daný vstup. Existuje viacero spôsobov, ako nájsť tzv. víťazný neurón. V našom prípade je to neurón, ktorého váhy sú najbližšie k aktuálnemu vstupu podľa euklidovskej vzdialenosti.

Po nájdení víťaza prichádza na rad učenie. V tomto kroku sa adaptujú váhy smerom k aktuálnemu vstupu. Neprenastavujeme však všetky váhy, ale len víťaza a jeho topologických susedov. Množinu susedov určuje funkcia okolia, pričom veľkosť okolia s časom postupne klesá (až kým medzi susedmi neostanú len najbližšie neuróny, príp. víťaz samotný). Zmena váh je ovládaná tzv. rýchlosťou učenia, ktorá tiež postupne klesá k 0.

Počas trénovania vieme obyčajne rozlíšiť dve fázy – usporiadavanie a doladenie. Vo fáze usporiadavania klesá veľkosť okolia diskrétne s časom. Vo fáze doladenia môžeme nechať v okolí už len najbližších susedov. Na presnom tvare funkcie poklesu rýchlosti učenia až tak nezáleží, musí to byť ale monotónne klesajúca funkcia od 1 na začiatku trénovania až po 0,1-0,01 vo fáze doladenia. Takisto nie je presne definovaný počet iterácií, mali by sme však dbať na to, aby sieť dokázala v prvej fáze nájsť správne globálne usporiadanie váhových vektorov, ktoré sa v druhej fáze už len lokálne doladia.

2.4.2 Perceptrón

Perceptrón je typ neurónovej siete trénovanej s učiteľom, ktorá má pevne stanovený počet vstupov a jeden výstup. Dá sa použiť na klasifikáciu vstupných vektorov do dvoch tried. Nevýhodou jednoduchého perceptróna je, že dokáže riešiť len tzv. lineárne separovateľné problémy ([10]). Vie sa teda naučiť klasifikovať len do takých tried, ktoré sú oddeliteľné nadrovinou v priestore definovanom vstupným vektorom. Tento nedostatok sa dá prekonať zavedením skrytej vrstvy. Tento typ siete budeme nazývať perceptrón so skrytou vrstvou.

Najčastejšie používaným učiacim pravidlom viacvrstvovej doprednej neurónovej siete (a teda aj perceptrónu so skrytou vrstvou) je algoritmus spätného šírenia chýb (*error backpropagation* alebo jednoducho *backpropagation*, pozri [11]). Váhy medzi výstupnou a poslednou skrytou vrstvou vieme meniť priamo podľa požadovaných výstupov. Zisťujeme pritom veľkosť chyby, ktorú potom ďalej šírime cez skryté vrstvy až k vstupu. Podrobný opis algoritmu spätného šírenia chýb môže čitateľ nájsť v [9].

2.4.3 Rekurentná neurónová sieť

Kým v doprednej neurónovej sieti sú povolené len spojenia v smere od vstupov k výstupom, rekurentná sieť môže mať aj prepojenia v rámci jednej vrstvy, či dokonca spätné prepojenia. To samozrejme zvyšuje jej možnosti uplatnenia, pričom ale musíme rátať aj so zložitejším trénovacím mechanizmom. Ak výstup siete nezáleží len od aktuálneho vstupu, ale aj od histórie doteraz predkladaných vzorov, rekurentná sieť si dokáže pomocou rekurentných prepojení zapamätať kontext, v akom sa vstup nachádzal, a tým ho asociovať so správnym výstupom.

Existuje viacero typov rekurentných neurónových sietí a príslušných trénovacích procedúr. Na tomto mieste opíšeme jednoduchú rekurentnú sieť so skrytou vrstvou, ktorá má rekurentné prepojenia na skrytej vrstve (známa ako Elmanova sieť). Tento typ siete vieme trénovať algoritmom spätného šírenia chýb v čase (*backpropagation through time*, pozri [12]). Sieť dostáva v trénovacej množine postupnosť vzorov, ktorú má asociovať s požadovaným výstupom. Postupnosť môže mať premenlivú dĺžku. Pri trénovaní sa sieť rozvíja v čase, pričom má toľko skrytých vrstiev, aká je dĺžka vstupnej postupnosti. Takto rozvinutú sieť následne trénujeme klasickým *backpropagation* algoritmom. Podrobnosti algoritmu môže čitateľ opäť nájsť v [9].

3 Mikrosvet

Pri modelovaní reálneho prostredia je veľmi náročné zachytiť a zahrnúť do modelu všetky relevantné informácie. Namiesto obmedzenia množstva spracovávaných znalostí o svete oklieštime samotné prostredie. Definujeme si jednoduchý mikrosvet, vďaka čomu presne poznáme všetky zákonitosti, ktoré v ňom platia.

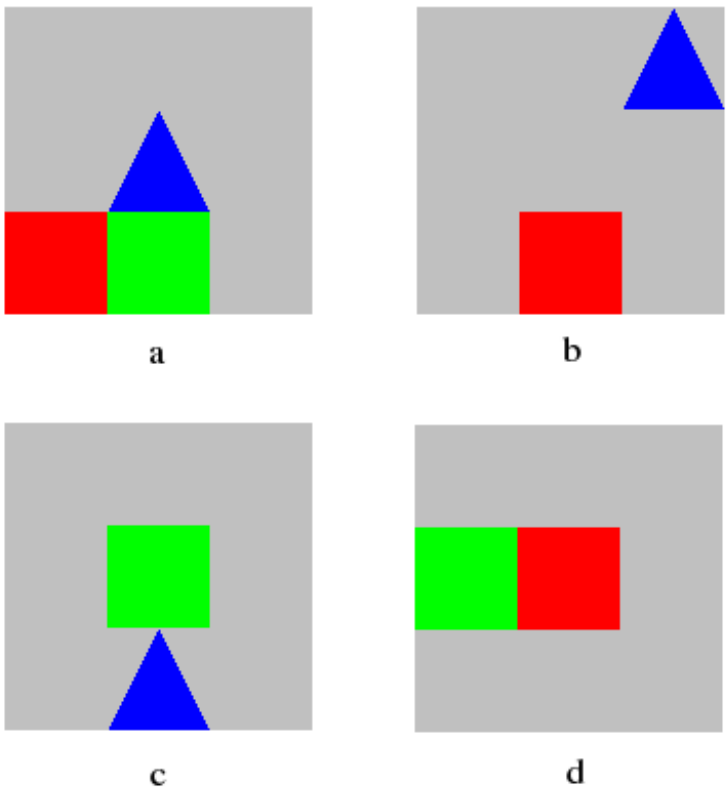
V tejto práci sme si zvolili prostredie pozostávajúce z geometrických objektov (blokov) rozmiestnených na dvojrozmernej mriežke. Podľa toho náš mikrosvet nazývame tiež *BlockWorld*. O porozumenie jazyka v podobnom prostredí sa snažil aj systém SHRDLU ([13]), svet je v ňom ale reprezentovaný na propozičnej úrovni.

V základnej verzii používame tri objekty – červený štvorec (*red square* resp. *red block*), zelený štvorec (*green square* resp. *green block*) a modrý trojuholník (*blue triangle* resp. *pyramid*). Mriežka má veľkosť 3×3 a môže obsahovať jeden až tri objekty (pozri obrázok 3.1). Objekty ale nemôžu byť rozmiestnené ľubovoľne. Vo svete funguje gravitácia – objekt môže byť položený buď na zemi, alebo na inom stabilnom objekte. Za stabilný objekt považujeme štvorec. Trojuholník je nestabilný, a teda nemôžeme priamo na neho postaviť iný objekt. K dipozícii máme aj pomyselné rameno žeriava, ktoré dokáže udržať objekt vo vzduchu. Mikrosvet môže obsahovať naraz najviac jeden objekt nachádzajúci sa vo vzduchu.

Naše prostredie má teda tieto základné obmedzenia:

- minimálne jeden a maximálne tri objekty na scéne
- gravitácia
- možnosť udržať maximálne jeden objekt vo vzduchu

Okrem toho máme ešte pravdepodobnostné obmedzenie, ktoré kontroluje počet situácií, v ktorých sa nachádza nejaký objekt vo vzduchu. Pravdepodobnosť generovania situácie z mikrosveta obsahujúceho objekt držaný ramenom žeriava je 0,5 (teda približne každá druhá inštancia z mikrosveta).



Obrázok 3.1.: Príklady rozmiestnenia objektov v mikrosvete. Obrázky (a), (b), (c) predstavujú možné stavy sveta; obrázok (d) nespĺňa podmienky mikrosveta. Na obrázku (a) nie je žiaden z objektov vo vzduchu. Na obrázku (b) je trojuholník držaný ramenom žeriava. Na obrázku (c) musí byť zelený štvorec držaný žeriavom, pretože trojuholník je nestabilný objekt, teda na neho nemôžeme postaviť iný objekt. Prípad (d) nespĺňa podmienku mikrosveta, že naraz môže byť najviac jeden objekt vo vzduchu, a preto nepredstavuje možnú inštanciu z mikrosveta.

Uvažujeme aj o možnosti modelovania mikrosveta väčších rozmerov. Prostredie sa

dá škálovať na viacerých dimenziách. Je to veľkosť mriežky, počet a typ objektov, množstvo objektov nachádzajúcich sa vo vzduchu. K téme škálovania mikrosveta sa vrátime v kapitole o situačnej reprezentácii.

3.1 Kódovanie mikrosveta

Existuje viacero spôsobov ako premeniť vizuálnu reprezentáciu mikrosveta na formu vhodnejšiu pre ďalšie spracovanie. Najjednoduchší spôsob je zaznamenať pre každý objekt jeho horizontálnu (os X) a vertikálnu (os Y) súradnicu. Keďže v našom svete sa nachádzajú tri objekty a má rozmery 3×3, potrebujeme šesť premenných s oborom hodnôt $\{0,1,2\}$. Označme ich `red_square_x`, `red_square_y`, `green_square_x`, `green_square_y`, `blue_triangle_x`, a `blue_triangle_y`.

Pri počítačovom spracovaní údajov je často vhodné, aby hodnoty dát boli v binárnej sústave. To dosiahneme tak, že pre každý objekt a každú súradnicu definujeme namiesto jednej premennej tri, jednu pre každú polohu. Teda napríklad namiesto `red_square_x` použijeme `red_square_x0`, `red_square_x1`, a `red_square_x2`. Pre každý objekt potrebujeme teraz šesť binárnych premenných, celú situáciu teda popíšeme pomocou 18 premenných.

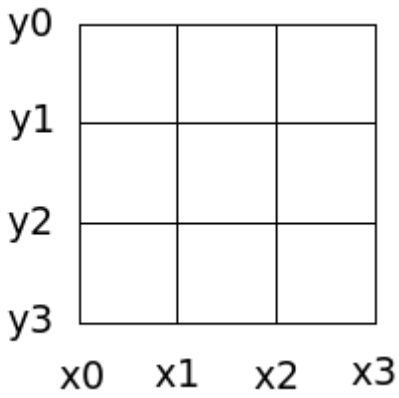
Nech sa napríklad na scéne nachádza červený štvorec na pozícii [0,2], zelený štvorec na pozícii [1,2] a modrý trojuholník na pozícii [1,1]. Aktívne premenné (nastavené na 1) pre túto situáciu budú `red_square_x0`, `red_square_y2`, `green_square_x1`, `green_square_y2`, `blue_triangle_x1` a `blue_triangle_y1`. Ostatné premenné budú neaktívne (rovné 0).

Takto získaná reprezentácia má ale jednu podstatnú nevýhodu – nie je v nej zachytená susednosť objektov. Tento nedostatok môžeme jednoducho prekonať tým, že každú polohu budeme kódovať dvoma premennými namiesto jednej. Jedna z premenných bude spoločná pre dve susedné pozície (pozri obrázok 3.1.1). Vzťahy podobnosti situácií na základe susednosti pozícií sú v tomto kódovaní zachytené. Keďže pozície sa v tomto kódovaní prekrývajú, nazýva sa toto kódovanie aj *coarse coding* (kódovanie s prekryvom).

Predstavme si, že sa objekt nachádza v ľavom dolnom rohu. Sú pre neho teda aktívne premenné `y2` a `y3` pre vertikálny smer a `x0` a `x1` pre horizontálny smer. Ak

objekt posunieme o jednu pozíciu doprava (bude teda v strede dole), vertikálna pozícia ostáva nezmenená a pre horizontálnu budú aktívne `x1` a `x2`. `x1` je aktívna v oboch prípadoch, čím je zachytená susednosť týchto pozícií.

Pre každý objekt teraz potrebujeme osem namiesto šiestich premenných, počet premenných na zakódovanie celého mikrosveta sa teda zvýšil na 24, čo ale môžeme tolerovať, keďže máme v kódovaní implicitne zahrnutú susednosť pozícií.



Obrázok 3.1.1.: Kódovanie pozícií objektov v mikrosvete.

Navrhnuté kódovanie má skutočný význam najmä pre mikrosvet väčších rozmerov, kde posun objektu o jednu príp. niekoľko málo pozícií nehrá podstatnú rolu.

Opísali sme si ako vyzerá náš mikrosvet a aké podmienky musí spĺňať. Navrhli sme taktiež kódovanie, ktoré použijeme pri budovaní situačnej reprezentácie. Môžeme teda prejsť k opisu samotnej reprezentácie.

4 Situačná reprezentácia

Cieľom tejto práce je otestovať možnosti situačnej reprezentácie na modelovanie sveta. Aby sme predišli problémom s nedostatočnými znalosťami o svete, zvolili sme si stratégiu opisu mikrosveta namiesto reálneho prostredia. Prvou úlohou je vytvoriť dostatočne robustnú situačnú reprezentáciu mikrosveta. Táto reprezentácia nemá byť založená na jazykovej či propozičnej charakteristike prostredia. Musí ale odrážať pravdepodobnosti výskytu jednotlivých situácií vo svete.

Situačná reprezentácia bude mať formu dvojrozmernej mapy kvôli dobrej vizualizácii. Chceme, aby spĺňala niektoré základné podmienky. Čím je pravdepodobnosť výskytu sveta väčšia, tým výraznejšia musí byť jemu prislúchajúca plocha na mape. Z toho priamo vyplýva, že keď máme špecifickejšiu informáciu o svete, plocha na mape sa zmenšuje. Požadujeme, aby reprezentácia konkrétnejšej scény bola podmnožinou reprezentácie všeobecnejšieho sveta.

4.1 Samoorganizujúca sa mapa

Ako už bolo spomenuté v prehľade, pre akékoľvek reálne použiteľné množstvo situácií nie je možné ručne vytvoriť situačnú reprezentáciu s požadovanými vlastnosťami. Dá sa to ale urobiť automaticky pomocou samoorganizujúcej sa mapy (SOM). Trénujeme ju na situáciách opísaných pomocou kódovania s prekryvom z predošlej kapitoly. Trénovacia množina pozostáva z 1000 náhodne vygenerovaných situácií. Tie sa môžu opakovať, odzrkadlujúc tak ich pravdepodobnosť výskytu.

Sieť má 15×15 buniek (neurónov) usporiadaných do hexagonálnej mriežky. Rozmery mapy boli zistené empiricky, pričom takáto veľkosť sa javí byť dostačujúca pre náš mikrosvet. Každý neurón má 24 váh zodpovedajúcich 24 premenným z použitého kódovania mikrosveta. V [2] každá váha zodpovedá jednej zo 14 základných propozícií opisujúcich situáciu v mikrosvete. My nemôžeme hovoriť o propozíciách, pretože náš mikrosvet nie je definovaný na báze propozícií, ale pomocou vyššie spomenutých premenných. Váhy SOM budeme označovať ako dimenzie, premenné, prípadne jednoducho váhy.

V sieti máme teda nasledovné dimenzie:

`red_square_x0`	`green_square_x0`	`blue_triangle_x0`
`red_square_x1`	`green_square_x1`	`blue_triangle_x1`
`red_square_x2`	`green_square_x2`	`blue_triangle_x2`
`red_square_x3`	`green_square_x3`	`blue_triangle_x3`
`red_square_y0`	`green_square_y0`	`blue_triangle_y0`
`red_square_y1`	`green_square_y1`	`blue_triangle_y1`
`red_square_y2`	`green_square_y2`	`blue_triangle_y2`
`red_square_y3`	`green_square_y3`	`blue_triangle_y3`

Každá váha nadobúda hodnoty medzi 0 a 1. Hodnota váhy v natrénovanej SOM naznačuje, do akej miery daný neurón reprezentuje objekt z mikrosveta nachádzajúci sa na danej pozícii. Frank túto hodnotu nazýva *membership value* a označuje $\mu_i(p)$ pre propozíciu *p*. U nás napríklad μ_i(`red_square_x0`) znamená, do akej miery i-ty neurón reprezentuje červený štvorec na pozícii *x0*.

Reprezentácia sveta je distribuovaná, objekt na jednej konkrétnej pozícii je teda opísaný viacerými neurónmi. Celková distribuovaná reprezentácia pre dimenziu *d* potom je $\mu(d)=(\mu_1(d),\mu_2(d),...,\mu_n(d))$, pričom *n* je v našom prípade rovné 225 (počet neurónov v SOM). Na reprezentáciu sa teda môžeme pozerať buď ako na dvojrozmernú plochu SOM alebo vektor, podľa toho, čo nám v danom prípade viac vyhovuje.

Už vieme ako vypočítať plochu SOM zodpovedajúcu jednej dimenzii. S tým si ale pri získavaní situačnej reprezentácie zodpovedajúcej inštanciám v *BlockWorld* nevystačíme, pretože v našom svete sú vždy naraz aktívne viaceré dimenzie. Konkrétne, ak sa vo svete nachádza len jeden objekt, na jeho charakterizáciu potrebujeme štyri dimenzie (ak sa napríklad červený štvorec nachádza vľavo dole, sú to dimenzie

`red_square_x0`, `red_square_x1`, `red_square_y2`, a `red_square_y3`). Pre dva objekty je to osem dimenzií, pre všetky tri objekty až 12 dimenzií.

Na získanie situačnej reprezentácie inštancie z mikrosveta preto potrebujeme zadefinovať operáciu konjunkcie pre dimenzie. Ako už bolo spomenuté, plocha SOM zodpovedajúca jednotlivým dimenziám nie je ostro definovaná, musíme preto využiť vzťahy z teórie fuzzy množín. V [2] je na výpočet konjunkcie použitý nasledovný vzorec:

$$\mu_i(p \wedge q) = \mu_i(p)\mu_i(q)$$

V našom mikrosvete ale tento výpočet konjunkcie nie je veľmi aplikovateľný, pretože vyžaduje prílišný počet násobení. Keďže váhy SOM nadobúdajú hodnoty medzi 0 a 1, násobením takýchto malých čísel vznikajú príliš malé výsledky. Získaná distribuovaná reprezentácia by potom bola veľmi nevýrazná. Frank tento spôsob výpočtu konjunkcie mohol použiť, pretože ním opisované situácie pozostávali z malého počtu základných propozícii (cca štyri propozície, čo sú len tri násobenia). My ale na opis sveta s tromi objektami, čo je typická situácia, potrebujeme až 12 dimenzií, čiže 11 násobení.

Navrhujeme preto iný spôsob výpočtu konjunkcie, tiež často využívaný vo fuzzy logike:

$$\mu_i(p \wedge q) = min\{\mu_i(p), \mu_i(q)\}$$

Hodnota na mape zodpovedajúca konjunkcii viacerých dimenzií je totožná s minimálnou z hodnôt pre tieto dimenzie. Je teda zrejmé, že aj pri aplikovaní konjunkcie na väčší počet dimenzií výsledná hodnota príliš neklesá.

Zatiaľ vieme, ako pre ľubovolnú inštanciu z mikrosveta získame jej distribuovanú situačnú reprezentáciu (vektor reálnych hodnôt medzi 0 a 1, prípadne plocha SOM). Budeme ale tiež potrebovať spôsob, ako z distribuovanej reprezentácie zistiť, akému rozloženiu sveta zodpovedá. Pre každú dimenziu vieme určiť presvedčenie (*belief value*), s akým táto dimenzia charakterizuje situáciu v mikrosvete.

Majme situačný vektor (alebo plochu SOM) $X = (x_1, x_2, \ldots, x_n)$. Tento vektor označuje situáciu *X*. Z vektora vieme určiť presvedčenie, s akým sa situácia *X* vyskytuje v mikrosvete. Zodpovedá mu časť plochy SOM, ktorú situácia pokrýva a vypočítame ho:

$$\tau(X) = \frac{1}{n}\sum_i x_i$$

Toto presvedčenie je približným vyjadrením subjektívnej nepodmienenej pravdepodobnosti, že sa situácia *X* vyskytne v mikrosvete. Pravdepodobnosť a presvedčenie nebudú totožné z dvoch dôvodov. Prvým je, že samotná reprezentácia získaná pomocou SOM je len aproximáciou mikrosveta. Druhý dôvod je priamym dôsledkom nami zvoleného kódovania mikrosveta. Dimenzie nie sú nezávislé. Dimenzie v strede (`x1`, `x2`) sú aktívne častejšie ako okrajové dimenzie (`x0`, `x3`), pretože každá z okrajových dimenzií slúži na kódovanie len jednej polohy, zatiaľ čo stredné dimenzie kódujú každá dve polohy. Presvedčenie sa teda nezhoduje s pravdepodobnosťou, platí tu ale vzťah, že čím je väčšie presvedčenie, tým je vyššia pravdepodobnosť (a naopak).

Na získanie informácie, akú reprezentáciu vektor zastupuje, potrebujeme poznať veľkosť presvedčenia, že je dimenzia *d* aktívna v situácii *X*. Označme toto presvedčenie $\tau(d|X)$. Táto hodnota aproximuje subjektívnu podmienenú pravdepodobnosť dimenzie *d* v situácii *X*. Pre podmienenú pravdepodobnosť platí nasledovné:

$$P(d|X)=\frac{P(d\wedge X)}{P(X)}$$

Po nahradení pravdepodobnosti (*P*) presvedčením (τ) a s využitím vyššie uvedených vzorcov pre výpočet konjunkcie a subjektívnej nepodmienenej pravdepodobnosti po úprave dostávame:

$$\tau(d|X)=\frac{\sum_i min\{\mu_i(d),x_i\}}{\sum_i x_i}$$

Pomocou tohto vzorca už dokážeme pre každú dimenziu ľahko určiť s akým presvedčením je aktívna v danej situácii.

Teraz už poznáme všetky potrebné vzorce na zistenie situácie z mikrosveta reprezentovanej situačným vektorom, preto môžeme uviesť algoritmus výpočtu pozície jednotlivých objektov. Pre každý objekt najskôr potrebujeme vedieť všetky dimenzie, ktoré môžu potenciálne určovať jeho pozíciu. Vypočítavame zvlášť horizontálnu a vertikálnu polohu. Horizontálna poloha červeného štvorca je definovaná dimenziami `red_square_x0`, `red_square_x1`, `red_square_x2`, a `red_square_x3`; vertikálna poloha dimenziami `red_square_y0`, `red_square_y1`, `red_square_y2`, a `red_square_y3`. Rovnako vieme určiť dimenzie pre zelený štvorec a modrý trojuholník. Sústreďme sa teraz na určenie horizontálnej polohy červeného štvorca. Pre

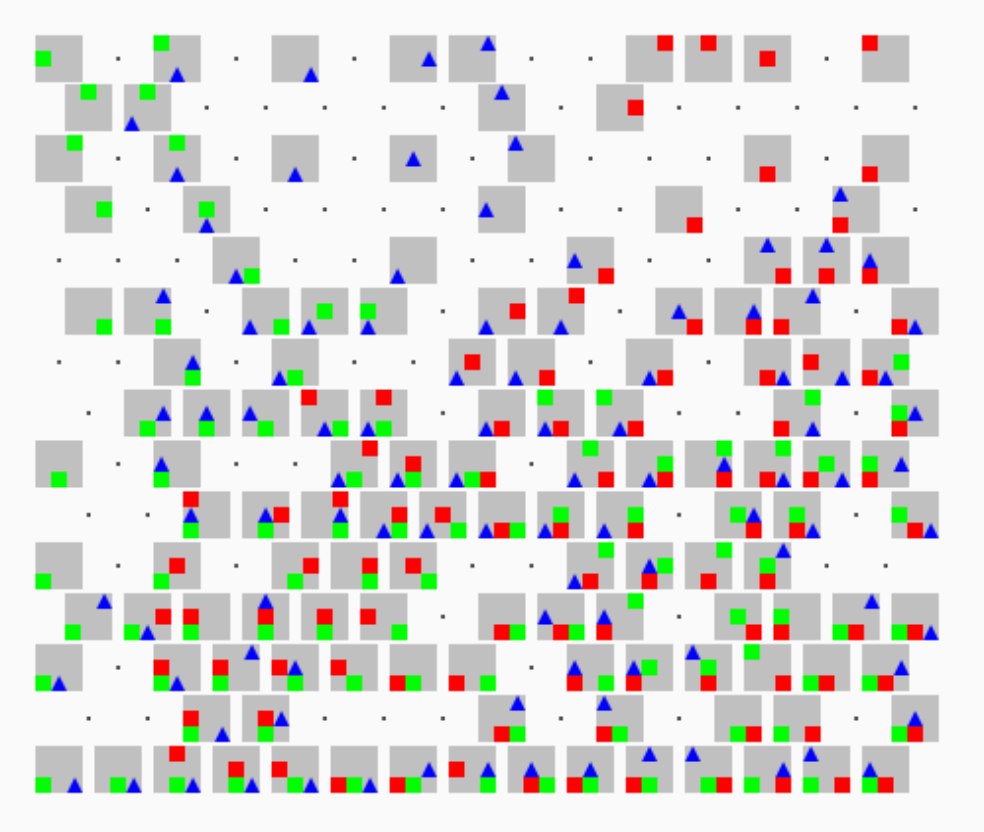
každú horizontálnu dimenziu vypočítame percentuálnu hodnotu presvedčenia, že táto dimenzia reprezentuje situáciu. Ak je presvedčenie o `red_square_x0` aj `red_square_x1` rovné 100%, horizontálnu polohu červeného štvorca nastavíme na 0. Ak to neplatí, skúšame postupne dvojice dimenzií `x1`, `x2` a `x2`, `x3`. V prípade, že nevieme presne určiť polohu nejakého objektu, predpokladáme, že tento objekt sa v situácii nenachádza.

Táto verzia algoritmu sa dá použiť len vtedy, ak máme k dispozícii presné hodnoty situačného vektora. Ak však máme len aproximáciu situačnej reprezentácie, presvedčenia o jednotlivých propozíciách tiež nebudú úplne presné (100%). V tom prípade dimenziu prijmeme, ak presahuje prahovú hodnotu. My sme túto hodnotu empiricky stanovili na 91%. Polohu potom určuje najvyššia zo súm `x0`, `x1` (poloha 0); `x1`, `x2` (poloha 1) a `x2`, `x3` (poloha 2). Okrem toho ešte uprednostňujeme okrajové pozície pred stredovou, pretože vo všeobecnosti majú presvedčenia spojené s dimenziami `x1` a `x2` vyššie hodnoty. Ak žiadna z dvojíc nemá obidve hodnoty nadprahové, objekt sa v situácii nenachádza.

4.2 Vizualizácia

Na natrénovanej SOM vieme zobraziť trénovacie vzory na mieste toho neurónu, ktorý tento vzor najviac vystihuje. Obrázok 4.2.1 ukazuje rozmiestnenie trénovacích vzorov po jednom zbehnutí trénovacej procedúry. Na obrázku nie sú vidno všetky prvky z trénovacej množiny, pretože jeden neurón môže byť najlepším reprezentantom pre viacero vzorov a zobrazený môže byť len jeden z nich.

Inštancie z mikrosveta nie sú na SOM rozmiestnené rovnomerne, ale je vidieť topografické usporiadanie situácií podľa ich vzájomných podobností. Sú badateľné oblasti, ktoré sú primárne zodpovedné za kódovanie umiestnenia jednotlivých objektov, a preto potrebujú viac miesta na mape. Umiestnenie červeného štvorca je primárne kódované pravým horným rohom SOM. Čiže ak sa vo svete nachádza červený štvorec, bude táto časť aktívna v jeho distribuovanej reprezentácii. Okrem toho bude aktívna ešte aj nejaká iná časť, kde je vidno inštancie zo sveta, ktoré obsahujú červený štvorec. Podobne za modrý trojuholník je primárne zodpovedná oblasť SOM hore v strede a zelený štvorec je kódovaný ľavým okrajom SOM.



Obrázok 4.2.1.: Zobrazenie situácií na natrénovanej SOM.

Na SOM ostávajú zachytené podobnosti vzorov aj v podrobnejšom meradle. Všimnime si napríklad ľavý dolný roh SOM. Spoločným znakom vzorov v tomto priestore je, že trojuholník je vpravo dole a zelený štvorec je hneď naľavo od neho. Nachádzajú sa tu kombinácie tohto sveta s rôznym umiestnením zvyšného červeného štvorca (a samozrejme aj bez červeného štvorca). Podobných oblastí dokážeme nájsť na mape hneď niekoľko. V podstate keď sa zameriame na ľubovoľný bod mapy, v jeho okolí sa nachádzajú jemu podobné svety.

Samozrejme, po opätovnom spustení trénovacieho procesu by natrénovaná SOM vyzerala odlišne, vedeli by sme v nej ale tiež odhaliť podobné zákonitosti ako v tomto prípade.

4.2.1 Distribuovaná reprezentácia

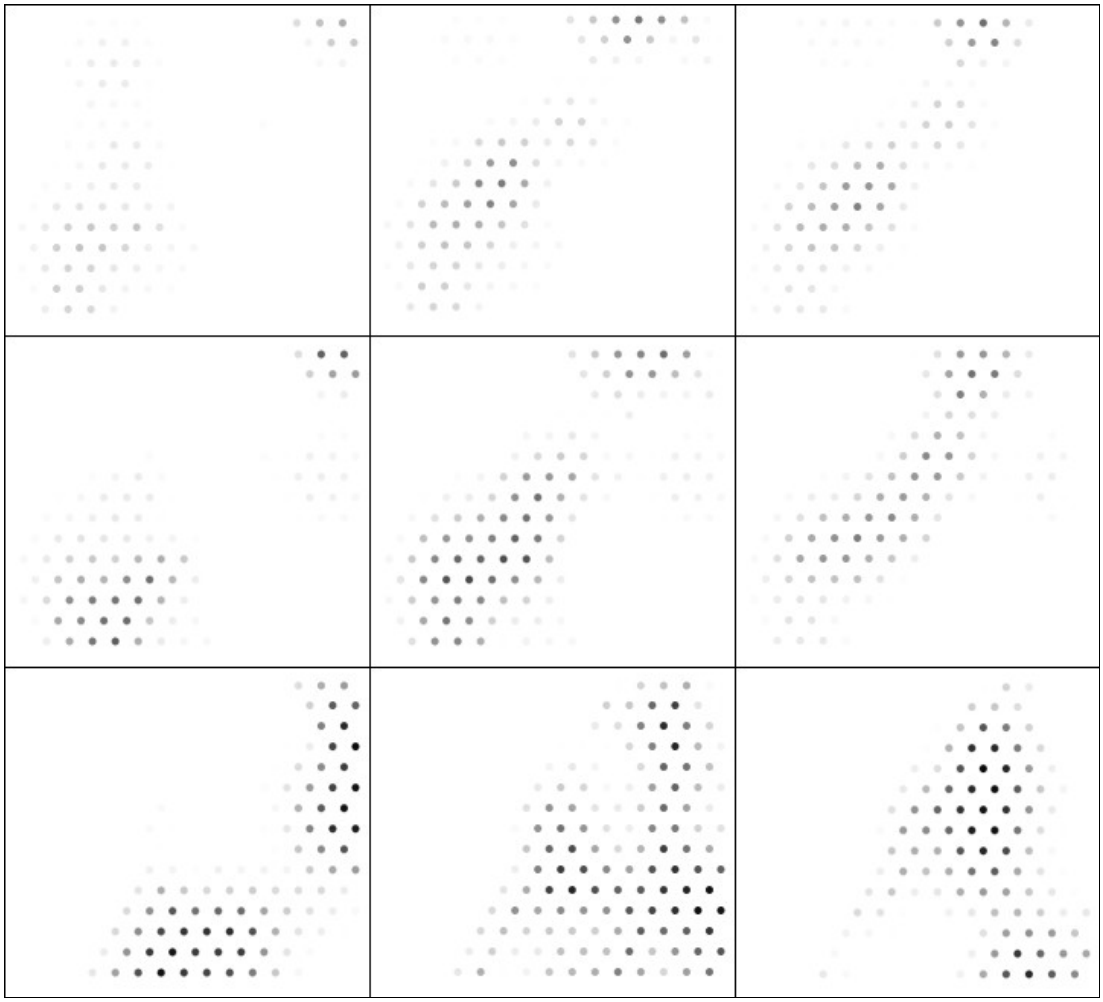
Zobrazenie jednotlivých inštancií z mikrosveta na mape predstavuje len lokalistickú reprezentáciu sveta, pretože pre každú inštanciu poznáme iba konkrétny

jeden neurón, ktorý ju najlepšie reprezentuje. Nás však zaujíma distribuovaná reprezentácia, teda miera reprezentácie mikrosveta každým neurónom. Tú vieme vypočítať z aktívnych dimenzií v mikrosvete a natrénovaných váh pomocou vzorcov z kapitoly o SOM.

Pozrime sa teraz bližšie na reprezentácie niektorých inštancií mikrosveta. Obrázok 4.2.1.1 znázorňuje distribuované reprezentácie červeného štvorca na rôznych pozíciách v mikrosvete. Aj voľným okom je z neho jasne viditeľná pravdepodobnosť výskytu jednotlivých inštancií mikrosveta. Reprezentácie v hornom rade sú celkom slabé. Plocha mapy, ktorú tieto reprezentácie zaberajú, je postupne 0,03; 0,06; a 0,055. Táto plocha zodpovedá presvedčeniu, že takáto situácia v mikrosvete nastane. Reprezentácie v strednom rade sú už o niečo výraznejšie. Plocha je postupne 0,065; 0,114; a 0,079. V dolnom rade sú reprezentácie najvýraznejšie. Plocha zodpovedá hodnotám 0,159; 0,215; a 0,16.

Tieto reprezentácie sú konzistentné s mikrosvetom, ktorý zachytávajú. Platí v ňom gravitácia, a teda objekt sa s najväčšou pravdepodobnosťou nachádza na zemi (spodný rad), prípadne (s menšou pravdepodobnosťou) na nejakom inom objekte. Okrem toho môže byť ešte objekt držaný ramenom žeriava – naraz ale môže byť držaný maximálne jeden objekt, čo znižuje pravdepodobnosť tohto javu.

Môžeme taktiež porovnať získané distribuované reprezentácie (Obrázok 4.2.1.1) s lokalistickým zobrazením inštancií mikrosveta (Obrázok 4.2.1). Pri opise lokalistickej reprezentácie sme uviedli, že červený štvorec je primárne kódovaný pravým horným rohom SOM. Ako vidíme, aj v distribuovaných reprezentáciách je táto časť mapy vždy aktívna. Okrem toho sú aktívne aj ďalšie časti, konkrétne tie, kde sa v lokalistickej reprezentácii nachádzajú inštancie mikrosveta obsahujúce červený štvorec. Časti mapy, ktoré primárne kódujú samostatný modrý trojuholník a zelený štvorec (oblasť hore v strede a ľavý okraj) nie sú v distribuovanej reprezentácii červeného štvorca zastúpené vôbec.

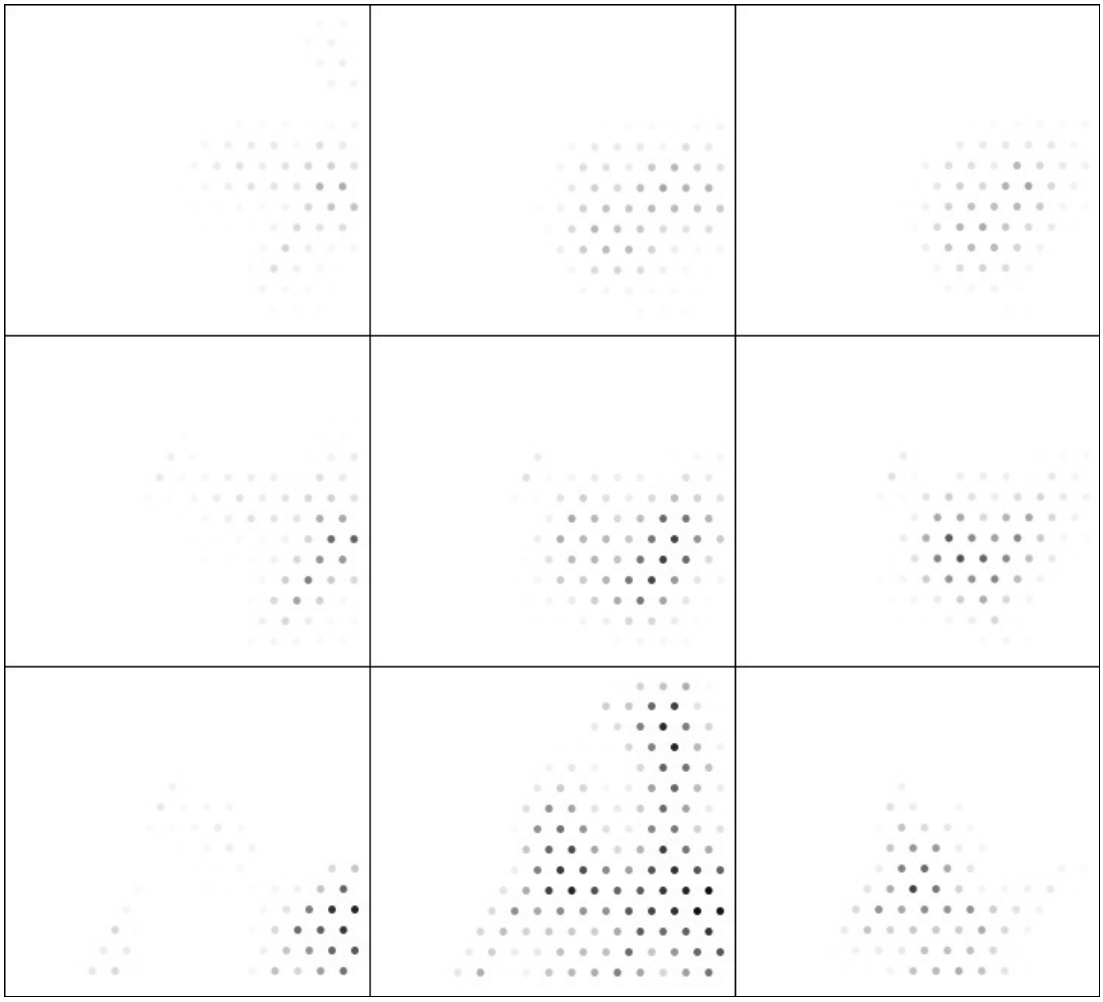


Obrázok 4.2.1.1.: Distribuované reprezentácie červeného štvorca na rôznych pozíciách v mikrosvete (pozícia reprezentácie na obrázku zodpovedá pozícii objektu vo svete).

Ešte zaujímavejšie je porovnanie distribuovaných reprezentácií viacerých objektov. Obrázok 4.2.1.2 znázorňuje situácie, na ktorých je zafixovaný červený štvorec v strede dole a zelený štvorec vystrieda všetky možné ostatné pozície vo svete. Dostaneme tak osem situácií, pre úplnosť uvádzame aj situáciu so samostatným červeným štvorcom.

Pri porovnaní distribuovanej reprezentácie samostatného červeného štvorca (v strede dole) s ostatnými reprezentáciami vidíme, že reprezentácia samého červeného štvorca je najvýraznejšia (čo je zrejmé), a že v sebe zahŕňa všetky ostatné reprezentácie. Táto kompozičnosť je jedna z vlastností, ktorú sme od dobrej situačnej reprezentácie mikrosveta požadovali. Táto vlastnosť by sa mala dať využiť pri postupnom verbálnom opise scény. Predstavme si, že chceme reprezentovať svet opísateľný vetou *červený*

štvorec je v strede dole a zelený štvorec je vpravo hore. Najskôr opíšeme všeobecnejšiu scénu (*červený štvorec je v strede dole*), a až následne ju bližšie špecifikujeme (... *a zelený štvorec je vpravo hore*). Reprezentácia sa tak môže budovať postupne, na viac krokov. Systém si ju začne vytvárať hneď potom ako získa informáciu o prvom objekte, a neskôr ju upresňuje ďalšími údajmi. Úpravy reprezentácie sú pritom konzistentné a spočívajú len v jej zjemňovaní a nie v celkovom pretváraní.



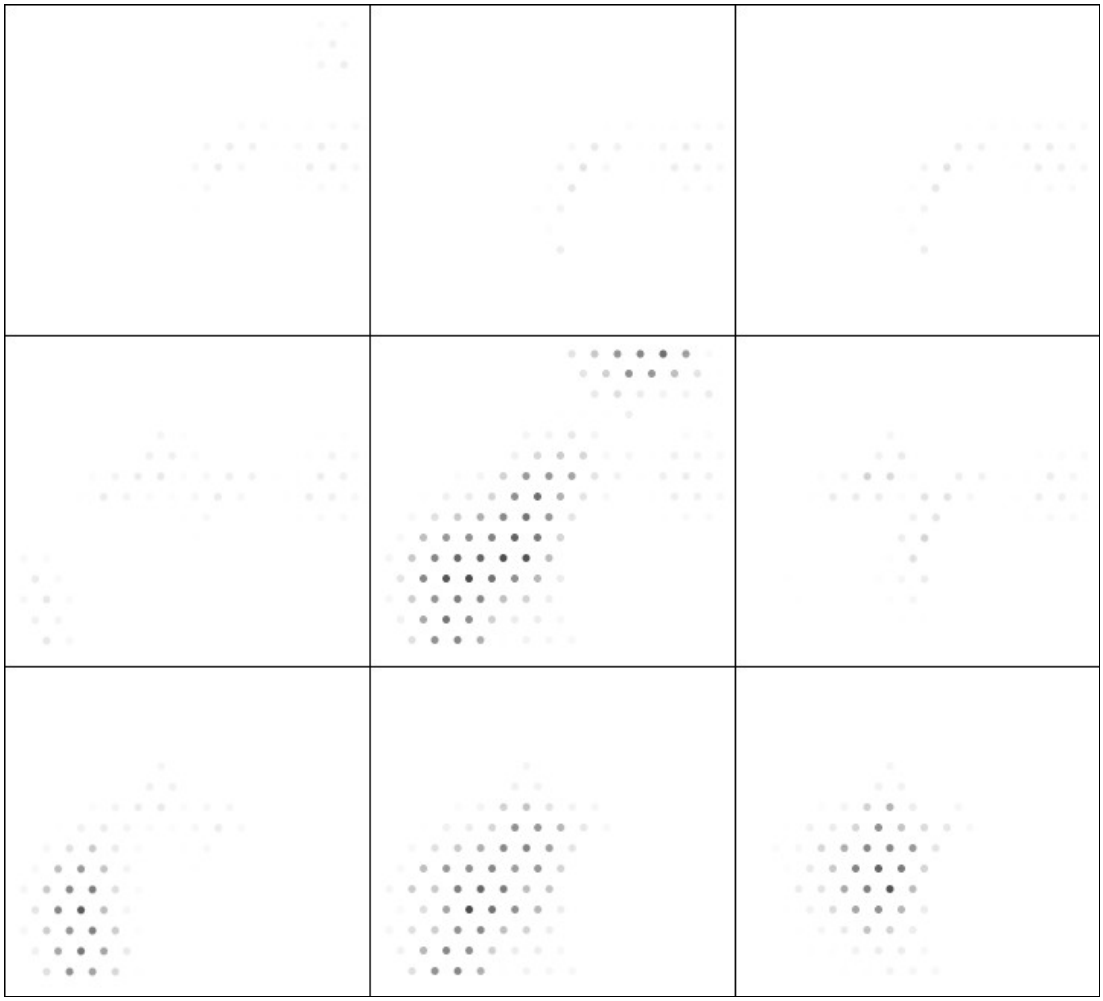
Obrázok 4.2.1.2.: Distribuované reprezentácie červeného štvorca na pozícii v strede dole so zeleným štvorcom na ostatných voľných pozíciách (pozícia reprezentácie na obrázku zodpovedá pozícii zeleného štvorca vo svete, v strede dole je reprezentácia samotného červeného štvorca, keďže dva objekty nemôžu byť naraz na tej istej pozícii).

Pri bližšom skúmaní reprezentácií zodpovedajúcich fixne umiestnenému červenému štvorcu a zelenému štvorcu inde vo svete si všimneme, že keď tieto

reprezentácie pospájame, získame takmer celú plochu opisujúcu samostatný červený štvorec v strede dole. Plocha, ktorú týmto pospájaním určite nedostaneme, je pravý horný roh mapy. Dôvod bol spomenutý už vyššie, táto časť mapy totiž reprezentuje samostatný červený štvorec a preto sa nezúčastňuje na opise viacerých (dvoch alebo troch) objektov.

Pozrime sa ešte na jednu množinu reprezentácií (obrázok 4.2.1.3). Je na nich opäť zafixovaný červený štvorec, tentokrát však presne v strede mriežky. Zelený štvorec, podobne ako v predošlom prípade, nadobúda všetky ostatné pozície, pričom reprezentácia v strede obrázku zodpovedá samotnému červenému štvorcu v strede plochy. Z obrázkov je opäť badateľná kompozičnosť vytvorenej situačnej reprezentácie – reprezentácia samotného červeného štvorca v sebe zahŕňa ostatné uvedené reprezentácie.

Táto množina reprezentácií je ale zaujímavá najmä z iného hľadiska. Vidíme, že reprezentácie v hornom rade, ako aj vľavo a vpravo v strede sú veľmi slabé, takmer nulové. Presvedčenia, že sa niektorá zo situácií v hornom rade vyskytne vo svete, sú postupne 0,004; 0,0042; a 0,004. Hodnoty pre situácie vľavo a vpravo v strede sú 0,0074 a 0,0093. Tieto hodnoty sú približne desaťkrát menšie ako príslušné hodnoty z obrázku 4.2.1.1. Takéto inštancie sveta teda nie sú vo vytvorenej situačnej reprezentácie dostatočne zachytené. V našom mikrosvete to ale nie je problém. Naopak, našim požiadavkám na reprezentáciu to vyhovuje, pretože to nie sú možné stavy sveta – obsahujú totiž dva objekty vo vzduchu, čo je porušením podmienok mikrosveta. Keďže sa takéto vzory nenachádzali v trénovacej množine pre SOM, natrénovaná SOM takýmto vzorom poskytuje iba minimálny nutný priestor, a tak venuje viac priestoru inštanciám mikrosveta, ktoré sa naozaj môžu objaviť.



Obrázok 4.2.1.3.: Distribuované reprezentácie červeného štvorca v strede mikrosveta so zeleným štvorcom na ostatných voľných pozíciách (pozícia reprezentácie na obrázku zodpovedá pozícii zeleného štvorca vo svete, presne v strede je reprezentácia samotného červeného štvorca, keďže dva objekty nemôžu byť naraz na tej istej pozícii).

4.3 Klasifikácia situácií z mikrosveta

V tejto podkapitole sa bližšie pozrieme na jednu klasifikačnú úlohu, ktorá spočíva v rozdelení situácií do dvoch tried. V jednej triede sú tie situácie, ktoré obsahujú nejaký objekt držaný ramenom žeriava, v druhej sú všetky objekty položené na zemi alebo na inom objekte.

Na tomto mieste je dobré pripomenúť, že vzory použité pri trénovaní SOM neobsahovali explicitnú informáciu, či sa v danej situácii nachádza držaný objekt. Nedá sa to určiť ani z pozícií jednotlivých objektov, ale len na základe vzájomných vzťahov. Ak sa nám podarí nájsť spôsob, ako len z hodnôt situačného vektora určiť, či daná situácia obsahuje alebo neobsahuje držaný objekt, bude to znamenať, že táto informácia je v situačnej reprezentácii nejakým spôsobom zachytená.

Na riešenie klasifikačnej úlohy použijeme jednoduchý, príp. dvojvrstvový perceptrón. Vstupná vrstva sa skladá z 225 neurónov s aktivačnou funkciou v tvare unipolárnej sigmoidy. Každý vstupný neurón dostane jednu hodnotu z 225-rozmerného situačného vektora. Keďže situácie klasifikujeme do dvoch tried, stačí nám jeden výstup. Výstupná hodnota 1 označuje situáciu s držaným objektom, 0 označuje situáciu bez držaného objektu. Ak naučíme perceptrón správne klasifikovať vzory, bude to znamenať, že triedy sú lineárne separovateľné. Ak nie, použijeme na riešenie úlohy silnejší nástroj, konkrétne doprednú sieť so skrytou vrstvou.

Aby naše výsledky odzrkadlovali skutočné vlastnosti situačnej reprezentácie a neboli len dôsledkom náhodne inicializovaných váh neurónovej siete, trénovaciu procedúru necháme zbehnúť 10 krát. V každom behu vytvoríme množinu 300 rôznych inštancií mikrosveta. Od neurónových sietí obyčajne požadujeme, aby dokázali správne klasifikovať nielen trénovacie dáta, ale aby dávali korektné výstupy aj pre vzory, ktoré počas trénovania nevideli (testovacie dáta). Tejto vlastnosti siete hovoríme generalizácia alebo zovšeobecnenie. Preto týchto 300 rôznych vzorov rozdelíme v pomere 9:1 (270 trénovacích a 30 testovacích vzorov).

Skúšali sme trénovať perceptrón, avšak bez väčšieho úspechu. Chyba na trénovacej množine síce klesala, avšak chyba na testovacej ostávala príliš vysoká. Pokúsili sme sa teda natrénovať na túto úlohu doprednú neurónovú sieť. Najskôr bolo treba určiť optimálny počet neurónov na skrytej vrstve. Po sérii pokusov sa ako najvhodnejší javil variant s 50 skrytými neurónmi. Zvolili sme plné prepojenie medzi vstupnou a skrytou, ako aj skrytou a výstupnou vrstvou (t.j. každý neurón s každým). V každom behu trénovacej procedúry sme nechali zbehnúť 2000 trénovacích iterácií, pričom rýchlosť učenia sme fixne nastavili na 0,1. V nasledovnej tabuľke vidíme výsledky trénovania:

Beh	**E_{train}**	**E_{test}**
1	0,047660	0,198755
2	0,042523	0,315893

3	0,047054	0,176963
4	0,048928	0,187448
5	0,045475	0,378190
6	0,048465	0,263075
7	0,047583	0,132027
8	0,051984	0,222061
9	0,038737	0,252887
10	0,046856	0,167279
priemer	**0,0465265**	**0,2294578**

V prvom stĺpci sa nachádza číslo behu trénovacej procedúry, v druhom je priemerná chyba na trénovacej množine a v treťom priemerná chyba na testovacej množine. Chybu pre jeden vzor vypočítame ako absolútnu hodnotu rozdielu požadovaného a skutočného výstupu výstupného neuróna pre tento vzor. Uvádzané chyby sú priemerom chýb pre celú trénovaciu resp. testovaciu množinu.

V poslednom riadku sú priemerné hodnoty z 10 behov trénovacej procedúry. Vidíme, že chyba na trénovacej množine bola v každom behu približne rovnaká, priemerná hodnota je 0,0465265; s čím môžeme byť spokojný. Je to dostatočne nízke číslo, trénovací proces teda nebol zastavený predčasne. Na testovacej množine chyba už viac kolísala a jej priemerná hodnota je 0,2294578; čo je dosť vysoké číslo vzhľadom na to, že maximálna veľkosť chyby môže byť 1 (v prípade, že neurón klasifikuje vzor do opačnej triedy).

Z priemernej chyby na testovacej množine môžeme usúdiť, že dopredná neurónová sieť so skrytou vrstvou je čiastočne schopná generalizácie. Trénovacie vzory dokáže vynikajúco klasifikovať, avšak približne pätinu až štvrtinu testovacích vzorov dáva do zlej triedy. Sieti sa teda nepodarilo nájsť ideálne nastavenie váh, s ktorým by správne kategorizovala všetky neznáme situácie.

4.4 Škálovanie mikrosveta

V našej práci sa primárne zaoberáme mikrosvetom veľkosti 3×3, ktorý obsahuje jeden až tri objekty, pričom jeden z nich môže (ale nemusí) byť vo vzduchu. Pre tento svet vytvárame situačnú reprezentáciu, ktorá má celkom sľubné vlastnosti. Otázkou ale je, či dokážeme použiť rovnaký postup na vytvorenie reprezentácie mikrosveta väčších

rozmerov. Zväčšíme preto nachvíľu náš mikrosvet. Svet bude mať rozmery 6×6 a môže naraz obsahovať jeden až šesť objektov, pričom na výber máme tentokrát z desiatich rôznych objektov. Vo vzduchu môžu byť maximálne dva objekty.

Keďže sme zväčšili mikrosvet (a teda aj množinu možných situácií), musíme zväčšiť aj SOM, ktorá slúži na vytvorenie situačnej reprezentácie, aby dokázala tieto sitácie zachytiť. Nastavujeme rozmery SOM na 30×30 (keďže sme zdvojnásobili výšku aj šírku mikrosveta, aplikujeme toto pravidlo aj na veľkosť SOM). Mapu trénujeme na 3000 náhodne vygenerovaných inštanciách z mikrosveta (môžu sa aj opakovať).

Kvalitu vzniknutej situačnej reprezentácie testujeme pomocou aplikácie *BlockWorldVisualisation* (pozri prílohu A). V tejto aplikácii dokážeme vygenerovať situáciu z mikrosveta spĺňajúcu jeho obmedzenia. Následne sa k nej vytvorí jej situačná reprezentácia a zrekonštruuje sa situácia. Porovnaním pôvodnej a zrekonštruovanej situácie získame chybovosť reprezentácie.

Dve situácie porovnávame tak, že porovnávame pozície jednotlivých objektov. Posun objektu o jednu pozíciu penalizujeme hodnotou 1, ak sa objekt v jednej z reprezentácií nenachádza (a mal by sa), penalizácia je 11. Táto hodnota nie je náhodná. Vypočítame ju *šírka + výška – 1*, aby bola táto hodnota väčšia ako maximálny posun objektu na ploche (päť pozícií horizontálne a päť vertikálne, spolu chyba 10).

Vypočítame priemernú chybu na 300 rôznych situáciách, pričom si necháme vygenerovať desať takýchto sád po 300 situácií:

Množina	**Chyba**
1	0,4
2	0,32
3	0,56
4	0,22
5	0,21
6	0,12
7	0,28
8	0,39
9	0,55
10	0,5
priemer	**0,355**

Výsledná priemerná chyba je 0,355; čo znamená približnú chybu o jednu pozíciu v každej tretej situácii. Keď uvážime, že mikrosvet je už väčší, tak v ňom posun objektu o jednu pozíciu príliš nezaváži. So schopnosťou rekonštrukcie teda môžeme byť spokojní. Treba si ale uvedomiť, že v tomto prípade sme rekonštruovali dokonalé situačné reprezentácie. Nevieme, ako úspešná by bola rekonštrukcia zašumených situácií. Bolo by tiež zaujímavé vyskúšať porozumenie viet opisujúcich takéto zložitejšie situácie. V nasledujúcej kapitole sa zaoberáme práve porozumením textu, avšak pracujeme len s pôvodným (jednoduchším) mikrosvetom.

5 Jazyk

Hlavným cieľom tejto práce je vyskúšať možnosti aplikácie situačnej reprezentácie v procese porozumenia jazyka. Zaoberáme sa zjednodušeným jazykom (mikrojazykom), opisujúcim situácie v mikrosvete. V našom modeli stotožňujeme porozumenie vety z jazyka s vytvorením vhodnej situačnej reprezentácie zodpovedajúcej danej vete. Vytvárame tak mapovanie symbolovej sekvencie (vety) na subsymbolovú situačnú reprezentáciu.

V tejto kapitole definujeme generátor jazyka, ktorý vyrába verbálny opis situácie v mikrosvete. Navrhneme tiež, ako možno riešiť problém porozumenia textu pomocou situačnej reprezentácie.

5.1 Generátor jazyka

Náš mikrojazyk obsahuje 15 slov, ktoré opisujú objekty na scéne, ich vzájomnú polohu, a slúžia tiež na spájanie vetných častí do viet či celých viet do súvetí. Používame nasledujúce slová: *red, green, block, pyramid, left, in_middle, right, up, bottom, and, is, just, of, on_top, above*.

Na scéne sa môžu nachádzať tri objekty. Červený štvorec označujeme *red block*, zelený štvorec je *green block* a modrý trojuholník je jednoducho *pyramid* (nepotrebujeme výraz pre modrú farbu, pretože trojuholník je len jeden, netreba ho teda bližšie špecifikovať).

Poloha môže byť definovaná buď absolútne (napr. *red block is up left*), alebo

relatívne, pomocou iného objektu (*pyramid is on_top of green block*). Pri generovaní viet uprednostňujeme relatívny opis pred absolútnym (samozrejme, aspoň jeden objekt na scéne musí byť opísaný v absolútnych súradniciach). Pri absolútnom opise používame výrazy *left, in_middle, right, up, bottom.* Termín *in_middle* označuje pozíciu v strede v horizontálnom aj vertikálnom smere. Vedeli by sme tento výraz nahradiť dvoma výrazmi reflektujúcimi odlišnosť medzi horizontálnym a vertikálnym stredom. Ponechanie jedného výrazu pre oba významy ale simuluje dobre známy jav z prirodzeného jazyka – homonymiu[1]. Môžeme teda pozorovať, ako sa s týmto javom porozumenie jazyka založené na situačnej reprezentácii vysporiada.

Ak sa objekt nachádza na zemi, vynechávame pri jeho opise informáciu o jeho vertikálnej pozícii (napr. namiesto *red block is bottom left* povieme len *red block is left*). Tým sa snažíme urobiť náš mikrojazyk trošku viac prirodzeným. Pre ľudskú reč je typické vynechávanie informácie, ktorá sa dá ľahko vydedukovať. Z pravidiel generovania inštancií mikrosveta vyplýva, že objekt bude s väčšou pravdepodobnosťou na zemi ako na vyšších pozíciách, preto túto informáciu vo vetách explicitne neuvádzame. Výraz *bottom* teda stráca svoje opodstatnenie, nechávame ho však v jazyku pre prípad, že by sme chceli generátor jazyka pozmeniť.

Pri relatívnom opise polohy vzťahujeme polohu objektu k referenčnému objektu. Ak majú opisovaný a referenčný objekt rovnakú vertikálnu súradnicu, polohu určíme výrazmi *left* alebo *right* (napr. *red block is right of pyramid*). Túto konštrukciu použijeme, ak sa medzi objektami nachádza voľné miesto prípadne iný objekt. Ak sa objekty nachádzajú tesne vedľa seba, polohu upresníme slovíčkom *just* (*red block is just right of pyramid*). V prípade, že sa rovnajú horizontálne súradnice objektov (objekty sú nad sebou), používame výrazy *on_top* a *above*. Výraz *on_top* opisuje objekty priamo položené na sebe (*pyramid is on_top of red block*), použitie termínu *above* naznačuje, že medzi objektami je medzera, prípadne ďalší objekt (*pyramid is above red block*).

Keď sa vo svete nachádza naraz viac objektov, scénu popíšeme jednoduchým súvetím. Vety spájame spojkou *and* (napr. *red block is left and pyramid is just right of red block*). Samozrejme, nie vždy sme schopní opísať polohu druhého a tretieho objektu relatívne, vtedy použijeme absolútnu pozíciu.

V našom modeli ku každej situácii existuje len jeden verbálny opis. Situácie by sa však často dali opísať viacerými spôsobmi. Napríklad význam vety *red block is left and*

1 Homonymá sú slová, ktoré rovnako znejú, ale majú odlišný význam.

pyramid is just right of red block je v našom mikrosvete ekvivalentný významu vety *pyramid is in_middle and red block is just right of pyramid*. Náš systém ale generuje (a teda aj následne rozpoznáva) len prvú z viet. Bolo by teda zaujímavé otestovať schopnosť porozumenia jazyka, ak by sa mohlo viac viet mapovať na tú istú situačnú reprezentáciu (*many-to-one mapping*). V tejto práci ale uvažujeme len o jednoznačných verbálnych opisoch sitácií (*one-to-one mapping*).

5.2 Porozumenie jazyka

Na porozumenie jazyka používame rekurentnú neurónovú sieť, ktorú trénujeme algoritmom spätného šírenia chýb v čase (*backpropagation through time, BPTT*). Sieť dostane na vstupe postupnosť slov, ktorá zodpovedá vete z jazyka. Slová sú kódované pomocou *one-hot* kódovania, čiže pre každé slovo máme práve jeden neurón. Sieť má teda 15 vstupných neurónov. Výstupná vrstva sa skladá z 225 neurónov, jeden pre každú hodnotu situačného vektora. Okrem toho má sieť ešte aj skrytú vrstvu. Jej veľkosť sme empiricky stanovili na 60 neurónov.

Sieť sa má naučiť priradiť k postupnosti slov (vete) správnu situačnú reprezentáciu. Skúšame dve trénovacie stratégie. V prvej chceme sieť naučiť *porozumieť* situáciám obsahujúcim jeden alebo dva objekty. Vygenerujeme si takéto situácie a rozdelíme ich na trénovaciu a testovaciu množinu. Pozorujeme, akú chybu má sieť pri spracovaní trénovacích a testovacích dát.

Druhá stratégia je o čosi sofistikovanejšia. Sieť chceme tentokrát naučiť priradiť správnu situačnú reprezentáciu ku všetkým možným situáciám z mikrosveta (t.j. situácie obsahujúce jeden, dva alebo tri objekty). Pri trénovaní postupujeme dvojfázovo – najskôr jej predkladáme len jednoduchšie situácie (jeden alebo dva objekty) a až neskôr ukazujeme všetky situácie (jeden, dva alebo tri objekty).

5.2.1 Trénovacia stratégia 1

Sieť trénujeme na situáciách obsahujúcich jeden alebo dva objekty. Vygenerujeme 150 rôznych inštancií z mikrosveta, ktoré rozdelíme v pomere 9:1 na trénovaciu a testovaciu množinu. Necháme zbehnúť 4000 trénovacích iterácií, pričom počas prvých 2000 iterácií je rýchlosť učenia 0,2; v druhej polovici rýchlosť učenia znížime na 0,1.

Podobne ako pri úlohe klasifikácie situácií z mikrosveta do dvoch tried (držaný

resp. nedržaný objekt), aj tu necháme trénovaciu procedúru zbehnúť viackrát s rôznymi trénovacími (a teda aj testovacími) dátami. Pozorujeme chybu na trénovacej a testovacej množine. Chybu pre jeden vzor vypočítame ako súčet absolútnych hodnôt rozdielov skutočného a požadovaného výstupu pre každý z 225 výstupných neurónov. V nasledujúcej tabuľke sú výsledné chyby na trénovacej a testovacej množine z piatich rôznych behov trénovacej procedúry:

Beh	$\mathbf{E_{train}}$	$\mathbf{E_{test}}$
1	1,69	5,75
2	1,78	9,97
3	1,5	12,33
4	1,72	8,87
5	1,64	9,74
priemer	**1,67**	**9,332**

Vidíme, že priemerná chyba na trénovacej množine je dosť malá, približne 1,67. Na testovacích dátach je priemerná chyba 9,332; čo už môže spôsobiť problém pri získavaní situačnej reprezentácie pre jednotlivé vety. Táto chyba nás ale v konečnom dôsledku nemusí až tak zaujímať, ak sa nám podarí aj napriek istému šumu v situačnom vektore zrekonštruovať správnu situáciu.

Urobíme rovnaké porovnanie situácií ako v kapitole o škálovaní mikrosveta. Pripomeňme si ho. Porovnávame pôvodnú situáciu (ktorej verbálny opis predstavuje vstup siete) so situáciou, ktorú získame z vygenerovanej situačnej reprezentácie. Každý posun objektu o jednu pozíciu si zaslúži jednotkovú penalizáciu. Ak sa objekt nachádzal v pôvodnej a nenachádza vo vygenerovanej situácii (alebo naopak), penalizujeme túto chybu siete hodnotou 5 bodov (opäť *šírka + výška – 1*). Vypočítame takúto priemernú chybu pre všetkých 153 situácií obsahujúcich jeden alebo dva objekty, označíme ju E_{sit}. V nasledujúcej tabuľke je približná hodnota chyby pre našich päť natrénovaných sietí:

Beh	$\mathbf{E_{sit}}$
1	1,27
2	1,2
3	1,16
4	0,83
5	1,07
priemer	**1,11**

Výsledná priemerná chyba je niečo vyše 1, čo predstavuje priemerný posun o jednu pozíciu v každej situácii. Jednoduché situácie (obsahujúce len jeden objekt) sa však sieť naučila takmer bezchybne, pri zložitejších situáciách (jeden alebo dva objekty) sa stane, že nejaký objekt vynechá či pridá.

5.2.2 Trénovacia stratégia 2

Sieť tentokrát trénujeme dvojfázovo. V tomto prípade aplikujeme tzv. *starting small* paradigmu ([14]), čiže sieti najskôr ukazujeme jednoduchšie situácie a až keď sa ich naučí, pristúpime aj k zložitejším vzorom.

Najprv sieti teda predkladáme trénovacie dáta opisujúce jednoduchšie situácie z mikrosveta (jeden alebo dva objekty). Trénovacia množina sa skladá zo všetkých 153 takýchto situácií, testovaciu množinu zatiaľ nemáme. Sieť trénujeme, až kým chyba neklesne na dostatočne nízku hodnotu. Keďže trénovací proces je dosť časovo náročná úloha (závislá od veľkosti trénovacej množiny a priemernej dĺžky vstupných postupností), uspokojíme sa s priemernou chybou blízkou 1. Chybu vypočítavame ako v predchádzajúcej trénovacej stratégii. Predpokladáme, že situačná reprezentácia je natoľko robustná, že ju mierne vychýlenie spôsobené touto chybou príliš neporuší.

V druhej fáze trénovania prichádzajú na rad aj zložitejšie situácie (jeden, dva alebo tri objekty). Ak by sme sieti predkladali len situácie s tromi objektami, mohla by zabudnúť, čo sa doposiaľ naučila. Dáta tentokrát rozdelíme na trénovaciu a testovaciu množinu, aby sme vedeli pozorovať chybu nielen na vzoroch, ktoré sieť videla, ale aj na dátach, ktoré počas trénovania nemala k dispozícii. Týmto vieme určiť schopnosť generalizácie siete. Vytvoríme 300 rôznych inštancií z mikrosveta a rozdelíme ich v pomere 9:1 (270 trénovacích vzorov a 30 testovacích). Po natrénovaní (niekoľko tisíc iterácií) je chyba na trénovacej množine približne 1,56 a na testovacej množine asi 3,9. Tieto hodnoty vyzerajú celkom sľubne.

Skúsme ale vypočítať E_{sit}. Priemerná chyba na všetkých situáciách s jedným alebo dvoma objektami je 0,69281, čo je ešte lepší výsledok ako pri použití predchádzajúcej trénovacej stratégie. Treba ale podotknúť, že sieť mala v prvej fáze trénovania k dispozícii všetky situácie s jedným alebo dvoma objektami, mohla sa ich teda všetky naučiť. Priemerná chyba na všetkých situáciách z nášho mikrosveta (t.j. jeden, dva alebo tri objekty) je ale až 3,204893, s čím žiaľ spokojní byť nemôžeme. Predstavuje to približnú chybu o 3 pozície v každej situácii.

Sieti sa síce celkom podarilo zovšeobecňovať, čo sa naučila z trénovacích dát (to naznačuje veľkosť chyby na testovacej množine), ale rekonštrukcia situácie z vygenerovaného situačného vektora je nepresná.

6 Záver

V práci sme skúmali možnosti situačnej reprezentácie pri opise mikrosveta obsahujúceho geometrické objekty. Najpr sme zadefinovali vhodné kódovanie, ktoré zachytáva podobnosť situácií. Z príkladov situácií sme následne pomocou samoorganizujúcej sa mapy vytvorili distribuovanú reprezentáciu mikrosveta. Zadefinovali sme vzorce, ako možno zo situácie získať jej situačnú reprezentáciu, a naopak, ako možno zo situačného vektora zrekonštruovať situáciu.

Preskúmali sme vlastnosti situačnej reprezentácie z viacerých perspektív. Najprv sme sa zamerali na vizuálne porovnanie vybraných situácií, pričom sme overovali, či vzniknutá reprezentácia spĺňa predkladané požiadavky. Potom sme testovali schopnosti situačnej reprezentácie pri riešení klasifikačnej úlohy, ktorá spočívala v rozdelení situácií do dvoch tried – situácie s držaným objektom a bez držaného objektu. Podarilo sa nám na danú úlohu natrénovať perceptrón so skrytou vrstvou, avšak generalizačná schopnosť siete na testovacích dátach bola menšia než sme očakávali. Skúmali sme tiež možnosť škálovania mikrosveta na svet väčších rozmerov. Ukázalo sa, že použitý spôsob vytvorenia situačnej reprezentácie by sa zrejme dal aplikovať aj na komplexnejšie situácie.

Nakoniec sme situačnú reprezentáciu použili pri porozumení jazyka. Vytvorili sme generátor viet opisujúcich situácie z mikrosveta. Na porozumenie jazyka sme použili jednoduchú rekurentnú neurónovú sieť trénovanú algoritmom spätného šírenia chýb v čase. Sieť dostáva na vstupe postupnosť slov opisujúcu situáciu a jej úlohou je vygenerovať príslušnú distribuovanú reprezentáciu. Porozumenie vety teda dávame do

súvisu s procesom tvorby situačnej reprezentácie.

Sieť trénujeme dvoma spôsobmi. Pri prvom trénovaní sa snažíme naučiť sieť porozumieť jednoduchším situáciám (jeden alebo dva objekty), čo sa nám aj celkom darí. Druhá trénovacia procedúra je rozdelená na dve fázy. Najskôr sieti ukazujeme len jednoduchšie situácie (ako v prvom prípade) a až keď sa ich naučí pridávame aj vety opisujúce zložitejšie situácie. Jednoduchším situáciám sieť porozumie ešte lepšie ako v prvom prípade, porozumenie zložitejších situácií by si ešte vyžiadalo ďalší výskum.

V budúcnosti by bolo dobré vrátiť sa k problému porozumenia zložitejších situácií. Po jeho vyriešení by sa mohlo pokračovať rozširovaním mikrosveta. Zaujímavé by tiež mohlo byť skúmanie situačných reprezentácií za sebou nasledujúcich situácií (časové závislosti).

Príloha A – Implementácia

Výsledky opísané v tejto práci pochádzajú z počítačového modelu *BlockWorld*. Keďže ide o prostredie so špeciálnymi vlastnosťami, ktoré sme si sami zadefinovali, bolo treba ho aj vlastnoručne implementovať. Väčšinu zdrojových kódov som napísal sám. Jediný projekt z nižšie uvedených, ktorý som sám nenaprogramoval, je `JavaSOM`, simulujúci samoorganizujúcu sa mapu (jeho zdrojové kódy som však pre svoje potreby tiež mierne modifikoval).

Všetky projekty som implementoval v programovacom jazyku JAVA pomocou voľne dostupného vývojového prostredia Eclipse.

Pri práci som vytvoril nasledujúce projekty:

- `BlockWorld`
- `JavaSOM`
- `WeightsMap`
- `BlockWorldVisualisation`
- `BlockWorldVerbalization`
- `nnsim`

Ku každému projektu spomeniem len najdôležitejšie triedy a metódy. Táto príloha rozhodne nie je kompletnou projektovou dokumentáciou, mala by však pomôcť čitateľovi ľahšie sa zorientovať v zdrojovom kóde. Projekty budem opisovať v poradí v akom vznikali.

BlockWorld

`BlockWorld` je základný projekt celej počítačovej simulácie. Hlavnou triedou projektu je rovnomenná trieda `BlockWorld`. Jedna inštancia triedy `BlockWorld` predstavuje jednu situáciu z mikrosveta. Celá situácia sa skladá z jednotlivých blokov, čo sú inštancie triedy `Block`. Každý blok si pamätá svoj typ (štvorec, trojuholník, atď.), farbu (červená, zelená, modrá, atď.), pozíciu. Vie tiež, či je stabilný (napr. trojuholník

nie je stabilný, ale štvorec áno) a či je práve v danej situácii držaný.

Na výrobu blokov slúži `BlockFactory`. Pomocou metódy `BlockFactory.NewBlock()` vieme získať náhodný blok, prípadne tejto metóde môžeme pomocou parametrov nastaviť, aký blok chceme (typ, farba, pozícia). Pri náhodnom generovaní blokov sa bloky vytvárajú podľa nastavení v definičnom XML pre *BlockWorld*. Nastavuje sa v ňom o.i. počet dostupných blokov, ako aj minimálny a maximálny počet objektov na scéne a maximálny počet držaných objektov.

Novú náhodnú situáciu vytvoríme v triede `BlockWorld` pomocou metódy `Generate()`. Ak chceme vytvoriť nejakú konkrétnu situáciu, uprednostníme metódu `Generate(ArrayList<Block>)`, ktorá naplní svet blokmi zo zoznamu. Trieda `BlockWorld` dokáže reprezentovanú situáciu nakresliť (`Draw()`), verbálne opísať (`Tell()`) aj pretransformovať do XML (`ToXml(Document)`). XML reprezentácia sa používa na vytvorenie situačného modelu.

Spúšťacou triedou projektu je `BlockWorldWindow`. Po spustení sa objaví okno, ktoré síce nie je príliš užívateľsky priateľné, dokážeme na ňom však zobraziť náhodnú inštanciu z mikrosveta, prípadne vygenerovať vopred definovaný počet situácií, ktoré poslúžia na vytvorenie situačnej reprezentácie (pomocou projektu `JavaSOM`).

JavaSOM

Na vytvorenie situačnej reprezentácie z množiny ukážkových situácií z mikrosveta používame samoorganizujúcu sa mapu (SOM). Neimplementoval som vlastnú SOM, ale používam projekt `JavaSOM`, ktorý bol vydaný s licenciou GNU GPL (*General Public Licence*). Projekt je možné voľne šíriť a modifikovať (sú dostupné aj zdrojové kódy). Domovská stránka projektu je http://javasom.sourceforge.net/.

Pôvodným výstupom siete bolo len topografické rozmiestnenie trénovacích vzorov po natrénovaní SOM. Ja som ale potreboval najmä natrénované váhy SOM, tak som si do siete doprogramoval aj tento výstup. Váhy natrénovanej mapy potrebujeme na vytvorenie situačnej reprezentácie.

WeightsMap

Projekt `WeightsMap` je ďalšou dôležitou zložkou implementácie. Inštancia triedy `WeightsMap` slúži na uloženie váh (*weights*) SOM, z ktorých už vieme priamo získať situačnú reprezentáciu. `WeightsMap` obsahuje zoznam vrcholov – `MapNode`. Každý vrchol si pamätá svoju pozíciu na mape, váhy zodpovedajúce jednotlivým dimenziám (24 dimenzií – `red_block_x0`, `red_block_x1`, atď.) a výstupnú hodnotu. Práve výstupná hodnota vrcholu predstavuje jeden prvok zo situačného vektora. V našej reprezentácii teda `WeightsMap` obsahuje presne 225 vrcholov. Pre i-ty vrchol (`MapNode`) vracia jeho metóda `GetMembershipValue(String dimensions)` hodnotu $\mu_i(d)$. Reťazec `dimensions` predstavuje konjunkciu dimenzií, pričom jednotlivé dimenzie oddeľujeme čiarkami.

Situačnú reprezentáciu nejakej inštancie z mikrosveta zostrojíme pomocou metódy `Generate(ArrayList<Block>)`, ktorej parametrom je zoznam blokov aktuálne prítomných v mikrosvete. Vytvorená situačná reprezentácia sa uloží aj do grafického súboru vo formáte SVG (*Scalable Vector Graphics*), ktorý môžeme zobraziť pri vizualizácii.

Pri rekonštrukcii pôvodnej situácie zo situačnej reprezentácie používame metódu `GetBelief(String dimensions)`, ktorá vráti veľkosť presvedčenia, že konjunkcia dimenzií (`dimensions`) je aktívna v danej situačnej reprezentácii. Je to vlastne hodnota $\tau(d|X)$ definovaná v časti o situačnej reprezentácii. Okrem toho vieme určiť aj presvedčenie, s akým sa táto situácia vyskytuje v mikrosvete ($\tau(X)$). Získame ho volaním funkcie `GetBelief()`.

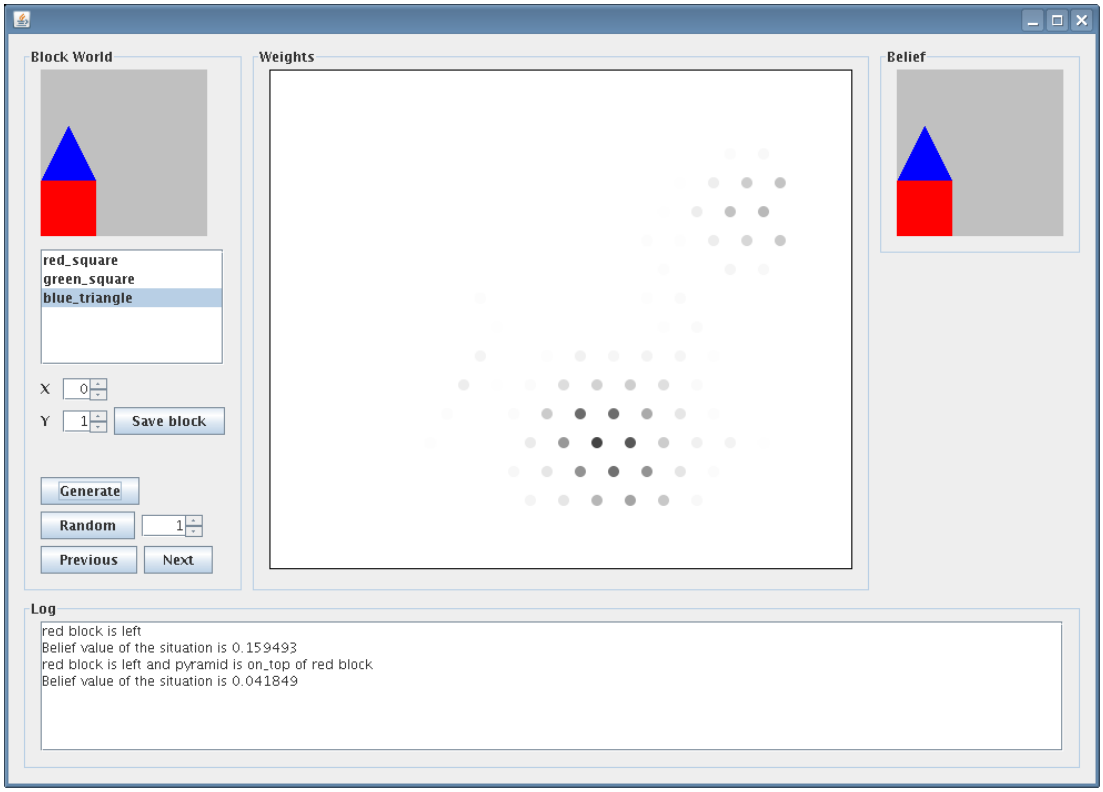
Distribuovanú situačnú reprezentáciu často potrebujeme nie ako dvojrozmernú mapu, ale ako vektor reálnych hodnôt. Na získanie tohto vektora použijeme metódu `GetDistributedRepresentation()` – vracia `ArrayList<Double>`.

BlockWorldVisualisation

Ako už názov napovedá, tento projekt slúži na vizualizáciu situácií z mikrosveta. Využíva pri tom už spomínané projekty `BlockWorld` a `WeightsMap`.

Na nasledovnom obrázku možno vidieť, ako vyzerá hlavné okno projektu. Teraz

bližšie popíšem jednotlivé vizuálne komponenty aplikácie.



Obrázok A.1.: Hlavné okno aplikácie BlockWorldVisualisation.

V ľavej časti okna sa nachádzajú ovládacie prvky umožňujúce nastaviť rozmiestnenie objektov v mikrosvete. Vykresľuje sa sem tiež aktuálna situácia. V strede je umiestnený panel zobrazujúci situačnú reprezentáciu v podobe dvojrozmernej mapy. Vpravo sa vykresľuje situácia zrekonštruovaná zo situačnej reprezentácie. V dolnej časti okna je textové pole, do ktorého sa vypisujú informácie pre používateľa.

Najviac pozornosti si zrejme zaslúži ľavý panel. Úplne hore plocha slúžiaca na vykreslenie aktuálnej situácie. Hneď pod ňou je komponent nazývaný *listbox*, z ktorého si používateľ môže vybrať jeden z ponúkaných objektov. Po výbere objektu mu môžeme nastaviť x-ovú a y-ovú pozíciu v políčkach vedľa písmen X a Y. Pozície sú číslované od 0, horizontálna zľava doprava, vertikálna zhora nadol. Ak nechceme nejaký objekt zahrnúť do situácie, nastavíme mu x-ovú aj y-ovú súradnicu na -1. Polohu každého objektu treba potvrdiť stlačením tlačidla *Save block*.

Po nastavení požadovaných pozícií objektov môžeme vygenerovať situáciu stlačením tlačidla *Generate*. Ak chceme vytvoriť náhodnú situáciu, stlačíme tlačidlo *Random*. Náhodná inštancia triedy `BlockWorld` sa opäť vytvára podľa definičného

XML súboru spomenutého v kapitole o projekte `BlockWorld`. Môžeme naraz vygenerovať aj viacero situácií, ich počet sa nastavuje naľavo od tlačidla *Random*.

Vygenerované situácie sa ukladajú do zoznamu. Tlačidlá *Previous* a *Next* slúžia na opätovné vykreslenie predchádzajúcej resp. nasledujúcej situácie zo zoznamu.

Po vygenerovaní situácie sa vykreslí jej situačná reprezentácia, z ktorej sa následne zrekonštruuje pôvodná situácia. Zrekonštruovaná sitácia sa vykreslí vpravo hore. Upozorňujem, že rekonštrukcia prebieha len na základe situačnej reprezentácie, pôvodná situácia sa do tohoto procesu vôbec nezapája.

Do textového poľa sa dajú vypísať rôzne informácie zaujímavé pre používateľa. V súčasnej verzii sa sem vypisuje verbálny opis vygenerovanej situácie, ako aj presvedčenie, s akým sa situácia vo svete vyskytuje.

Pri generovaní situácie sa okrem toho ešte aj ukladajú situácie ako vzory pre trénovanie klasifikácie situácií (držaný resp. nedržaný objekt). Ukladajú sa trénovacie dáta (súbor *hold.train.xml*), pričom každá desiata situácia sa uloží medzi testovacie dáta (*hold.test.xml*). Rovnako sa ukladajú aj vzory pre trénovanie a testovanie porozumenia jazyka (súbory *lang.train.xml* a *lang.test.xml*).

Vstupom pre aplikáciu je už spomínaný definičný súbor mikrosveta (*BlockWorldDefinition.xml*) a súbor s natrénovanými váhami SOM (*weigths.xml*), využívaný pri vytváraní situačnej reprezentácie a rekonštruovaná situácie.

nnsim

Projekt `nnsim` tu predstavím len v krátkosti, pretože jeho vytvorenie nebolo cieľom tejto práce, ale len prostriedkom na dosiahnutie cieľov. Je to simulátor umelej neurónovej siete. V súčasnej verzii pomocou neho dokážeme simulovať rekurentnú neurónovú sieť a doprednú neurónovú sieť. Rekurentnú sieť môžeme trénovať pomocou algoritmu spätného šírenia chýb v čase (*backpropagation through time*), doprednú sieť trénujeme klasickou metódou spätného šírenia chýb (*error backpropagation*).

Projekt je členený na balíčky `networks` (siete `RecurrentNeuralNetwork`, `NeuralNetwork` a generátor sietí `NeuralNetworkFactory`), `layers` (vrstvy), `neurons` (neuróny), `synapses` (prepojenia medzi neurónmi) a `patterns` (vzory).

Sieť vytvoríme z definičného XML súboru pomocou metódy

`NeuralNetworkFactory.loadNetwork(String fileName)`. Definícia siete môže vyzerať nasledovne:

```
<NeuralNetwork type="recurrent" name="lang">
     <Layers>
          <Layer type="input" name="input" size="15" />
          <Layer type="hidden" name="hidden" size="60" />
          <Layer type="output" name="output" size="225" />
     </Layers>
     <Connections>
          <Connection src="input" dest="hidden" />
          <Connection src="hidden" dest="hidden" />
          <Connection src="hidden" dest="output" />
     </Connections>
     <Settings>
          <TrainingSet filename="lang.train.xml" />
          <TestingSet filename="lang.test.xml" />
          <SaveWeights filename="lang.weights_out.txt" />
          <MaxIterations>1000</MaxIterations>
          <LearningRate>0.2</LearningRate>
     </Settings>
</NeuralNetwork>
```

V tomto súbore je zadefinovaná rekurentná neurónová sieť s názvom *lang*. Obsahuje 15 vstupných, 60 skrytých a 225 výstupných neurónov. Nasledujúce vrstvy sú plne prepojené: vstupná so skrytou, skrytá sama so sebou (rekurentné spojenia) a skrytá s výstupnou. Trénovacia množina sa načítava zo súboru *lang.train.xml* a testovacia z *lang.test.xml*. Výsledné váhy po natrénovaní siete sa uložia do súboru *lang.weights_out.txt*. Trénovací proces má zbehnúť maximálne 1000 iterácií s rýchlosťou učenia 0,2.

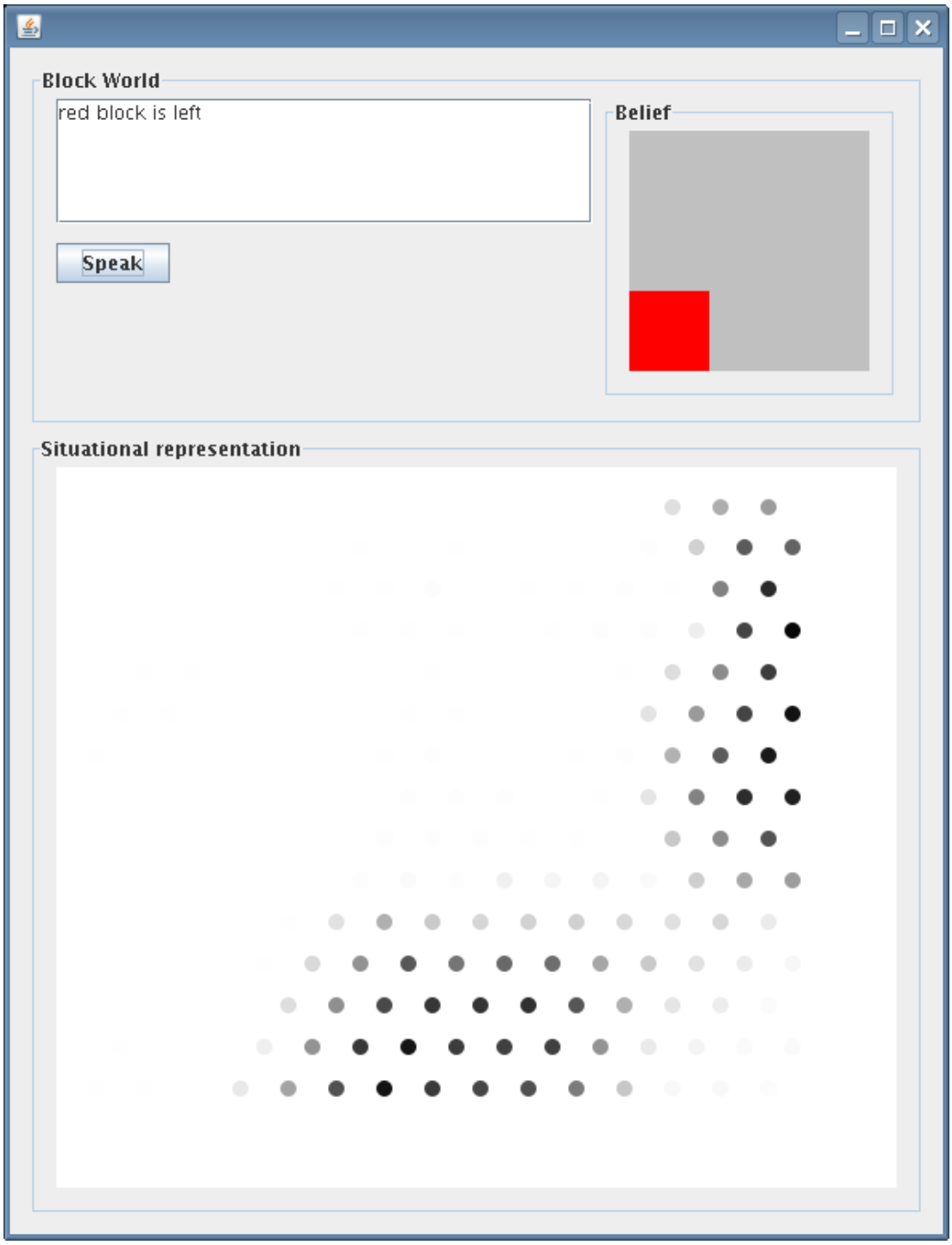
Po vytvorení siete sa môže začať trénovanie. Spúšťa sa metódou `run()` triedy `NeuralNetwork` (`RecurrentNeuralNetwork` je jej potomkom).

Ak máme už natrénovanú sieť a chceme len vypočítať výstup pre nejaký vstupný vzor, musíme po vytvorení siete načítať súbor s váhami prepojení. Na to slúži metóda `loadWeights(String fileName)`. Výstup pre konkrétny vzor (*pattern*) získame pomocou `testAndSaveOutput(Pattern pattern)`.

Viac o simulátore neurónových sietí `nnsim` môže záujemca nájsť v priložených zdrojových kódoch.

BlockWorldVerbalization

Táto aplikácia slúži na testovanie porozumenia jazyka. Hlavné okno aplikácie je na nasledovnom obrázku:



Obrázok A.2.: Hlavné okno aplikácie BlockWorldVerbalization.

Užívateľ napíše do textového poľa verbálny opis situácie (v našom prípade veta *red block is left*). Po stlačení tlačidla *Speak* sa najprv vygeneruje situačná reprezentácia zodpovedajúca našej vete a pomocou nej sa zrekonštruuje situácia. Ako vidíme, v tomto prípade dopadla rekonštrukcia výborne.

Porozumenie vety prebieha nasledovne. Najprv sa zo slov vytvorí vstupná reprezentácia pre rekurentnú neurónovú sieť. Vypočíta sa výstup natrénovanej siete pre túto vstupnú sekvenciu, čím získame situačnú reprezentáciu vety. Z nej už pomocou projektu `WeightsMap` vieme zrekonštruovať pôvodnú situáciu (to sme už potrebovali aj v projekte `BlockWorldVisualisation`). Zrekonštruovaná situácia sa nakoniec vykreslí.

Aplikácia `BlockWorldVerbalization` na svoj beh potrebuje súbor *weights_in.txt* s natrénovanými váhami neurónovej siete (na získanie distribuovanej reprezentácie situácie) a súbor *weights_som.xml* obsahujúci váhy natrénovanej SOM (na rekonštrukciu situácie).

Príloha B – Obsah CD

Na priloženom CD sa nachádza táto práca v elektronickej podobe, zdrojové kódy použitých aplikácií a výsledky experimentov.

V adresári *doc* je uložená elektronická verzia diplomovej práce vo formátoch *odt* (*OpenOffice Writer*) a *pdf* (*Portable Document Format*).

Do adresára *src* som nakopíroval pracovné prostredie z Eclipse, v ktorom som naprogramoval projekty opísané v prílohe o implementácii. Aplikácie sa dajú priamo použiť, sú nastavené na prácu s našim štandardným mikrosvetom. Okrem binárnej verzie sú priložené aj zdrojové kódy.

Nakoniec, v adresári *results* sú umiestnené výsledky našich experimentov. Podadresár *som* obsahuje súbor s váhami natrénovanej SOM, ktorý sa priamo používa na vytvorenie situačnej reprezentácie a ktorý sme využívali v našej práci. Sú tu nastavenia na trénovanie SOM aj ostatné výstupy vygenerované počas trénovania.

Podadresár *hold* obsahuje nastavenia perceptrónu použitého pri klasifikačnej úlohe (držaný alebo nedržaný objekt). Je tu aj desať rôznych variantov trénovacej a testovacej množiny spolu s príslušnými naučenými váhami.

V podadresári *scaling* sú logovacie súbory s porovnaniami pôvodnej a zrekonštruovanej situácie, ktoré sú použité v kapitole o možnostiach škálovania mikrosveta.

Podadresár *lang* obsahuje výsledky trénovania porozumenia jazyka. Trénovacie a testovacie dáta spolu s naučenými váhami neurónovej siete sú rozdelené do podadresárov podľa použitej trénovacej procedúry.

Referencie

[1] Zwaan, R.A., Radvansky, G.A. (1998). *Situation models in language comprehension and memory*. Psychological Bulletin, 123, 162-185.

[2] Frank, S.L., Koppen, M., Noordman, L.G.M., Vonk, W. (2003). *Modeling knowledge-based inferences in story comprehension*. Cognitive Science, 27, 875-910.

[3] Frank, S.L. (2005). *Sentence comprehension as the construction of a situational representation: a connectionist model*. In A. Russell, T. Honkela, K. Lagus, & M. Pöllä (Eds.), Proceedings of AMKLC'05. Espoo, Finland: Helsinki University of Technology.

[4] Johnson-Laird, P.N. (1983). *Mental Models: Towards a Cognitive Science of Language, Inference, and Consciousness*. Cambridge, MA: Harvard University Press.

[5] van Dijk, T.A., Kintsch, W. (1983). *Strategies in Discourse Comprehension*. New York: Academic Press.

[6] Schneider, W., Körkel, J. (1989). *The knowledge base and text recall: Evidence from a short-term longitudinal study*. Contemporary Educational Psychology, 14, 382-393.

[7] Frank, S.L., Koppen, M., Noordman, L.G.M., Vonk, W. (). *World knowledge in discourse-comprehension models*.

[8] Kohonen, T. (1995). *Self-Organizing Maps*. Berlin: Springer Verlag.

[9] Návrat et al. (2002). *Umelá inteligencia*. Vydavateľstvo STU, Bratislava.

[10] Minsky, M., Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. Cambridge, MA: MIT Press.

[11] Rumelhart, D.E., Hinton, G.E., Williams, R.J. (1986). *Learning Internal Representations by Error Propagation*. In McClelland, J.L., Rumelhart, D.E., and PDP Research Group, Parallel Distributed Processing: Explorations in the Microstructure of Cognition. MIT Press, Cambridge.

[12] Werbos, P.J. (1990). *Backpropagation Through Time: What It Is and How to Do It*. Proceedings of the IEEE, 10, 1550-1560.

[13] Winograd, T. (1977). *Five Lectures on Artificial Intelligence*. In Zampolli, A. (Ed.), Linguistic Structures Processing

[14] Elman, J.L. (1993). *Learning and Development in Neural Networks: The Importance of Starting Small*. Cognition, 48.