Het Categoriseren van Sequenties in een Neuraal Netwerk

Sophie J.I.M. Spitters (s1020986)

Universiteit Twente

Bachelorthese Human Factors en Mediapsychologie

1e begeleider: Prof. Dr. Frank van der Velde

2e begeleider: Prof. Dr. Ing. Willem Verwey

13 januari 2013

Samenvatting

In dit onderzoek is nagegaan hoe het categoriseren van sequenties kan worden gemodelleerd. Hiertoe zijn richtlijnen geformuleerd die het model moet volgen om de cognitieve processen categoriseren en sequentieleren te modelleren. Uit literatuuronderzoek blijkt het connectionisme een geschikte en biologisch plausibele benadering om dergelijke cognitive processen te modelleren. Dit heeft geleid tot het modelleren van het categoriseren van sequenties in een neuraal netwerk. Het neurale netwerk gebruikt voor dit onderzoek was een integratie van het *feedforward network* van Rumelhart, een model van categoriseren, en het *simple recurrent network* van Elman, een model van sequentieleren. Om te testen of het netwerk inderdaad in staat was sequenties te categoriseren, moest het netwerk routes leren en categoriseren naar draairichting. Uit de resultaten kan ten eerste worden geconcludeerd dat het netwerk tot op zekere hoogte in staat is de sequenties te leren. Daarnaast lijkt het netwerk in staat de geleerde sequenties te categoriseren. Bij deze conclusie moet rekening worden gehouden met het exploratieve karakter van het onderzoek.

Trefwoorden: categoriseren, sequentieleren, connectionisme, neuraal netwerk

Abstract

In this research the categorization of sequences has been modeled. For this purpose, guidelines have been formulated that need to be followed to model categorization and sequence learning in a biologically plausible way. Literature research states that the connectionist approach applies well to modeling such cognitive processes. Hence, categorizing sequences is modeled in an artificial neural network. The network used in this research is an integration of the Rumelhart feedforward network and the Elman simple recurrent network. To test if the network succeeded in categorizing sequences the network had to learn different routes and categorize them by rotation. Two conclusions follow from the results. First, the network is able to learn the sequences to a certain extent. Second, the network seems able to categorize the learned sequences. The fact that this is an explorative research should be taken into account while reviewing the conclusions.

Keywords: categorization, sequence learning, connectionism, artificial neural networks

Inhoudsopgave

1. Inleiding

Het categoriseren van informatie en het leren van sequenties zijn twee cognitieve processen waar al veel onderzoek naar is gedaan (bijv. Clegg, DiGirolamo, & Keele, 1998; Mandler, 2000; Pauen, 2002) . Hoe deze processen samenwerken is echter nog nauwelijks onderzocht. Dit onderzoek probeert hier meer inzicht in te krijgen door het categoriseren van sequenties te modelleren in een neuraal netwerk. Het modelleren van het categoriseren van sequenties zal worden gedaan door reeds bestaande modellen van categoriseren en sequentieleren te integreren. Het belang van dit onderzoek is ten eerste dat het onderzoek tot meer inzicht in de werking van het categoriseren van sequenties in de hersenen kan leiden. Daarnaast kan het integreren van twee bestaande modellen tot nieuwe functionaliteit leiden die die kan worden gebruikt in praktische toepassingen.

Om tot een geschikt model van het categoriseren van sequenties te komen, zijn eerst richtlijnen voor dit model geformuleerd. Deze richtlijnen komen voort uit de beschikbare kennis in de psychologie over categoriseren en over het leren van sequenties. Vervolgens wordt een model van categoriseren (Rogers & McClelland, 2008) en een model van sequentieleren (Elman, 1991) beschreven die deze richtlijnen grotendeels volgen. Beide modellen zullen worden geïntegreerd tot een model voor het categoriseren van sequenties. Dit model zal vervolgens getest worden om na te gaan of de modellen van categoriseren en sequentieleren inderdaad verenigbaar zijn en onder welke condities dit het geval is. Hierbij is rekening gehouden met het exploratieve karakter van dit onderzoek.

1.1. Categoriseren

De mens is bijzonder goed in het scheppen van orde in zijn chaotische omgeving. Verschillende objecten en concepten worden van elkaar onderscheiden en gegroepeerd in categorieën. Zo verschilt een hamer van een nijptang, maar vallen ze beiden onder de categorie gereedschap. Categoriseren is het cognitieve proces dat hieraan ten grondslag ligt.

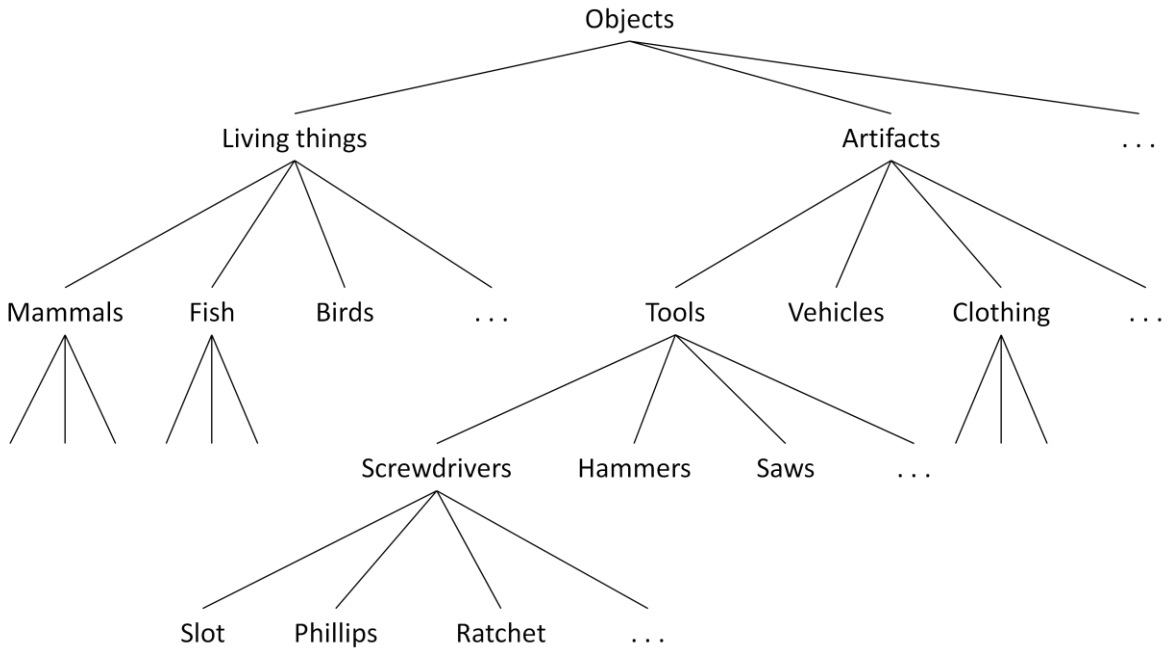
Het wordt als volgt gedefinieerd door Smith & Kosslyn (2009): *"Categorization is the ability to establish that a perceived entity belongs to a particular group of things that share key characteristics" (p. 149).*

Categoriseren is voor mensen van belang, omdat het hen in staat stelt informatie die niet expliciet aanwezig is over een bepaald concept af te leiden (Smith, & Kosslyn, 2009). Deze informatie is af te leiden aan de hand van kennis die men heeft over de categorie waartoe het concept hoort. Kennis van een categorie komt tot stand door eerst representaties te creëren van de verschillende leden van een categorie. Vervolgens worden deze representaties geïntegreerd tot representaties van de categorie.

Bij het categoriseren van concepten doen zich fenomenen voor die typisch zijn voor dit proces. In de volgende paragrafen worden een aantal belangrijke fenomenen van categoriseren besproken, namelijk: het proces van progressieve differentiatie, het bestaan van prototypes en de mogelijkheid tot een dynamische representatie van categorieën. De paragrafen worden afgesloten met richtlijnen voor het voorgestelde model van categorisatie. De richtlijnen volgen uit de bespreking van de fenomenen.

1.1.1. Progressieve differentiatie.

In het geheugen worden verschillende categorieën niet geïsoleerd gerepresenteerd, maar worden ze aan elkaar gerelateerd via verschillende structuren (Smith & Kosslyn, 2009). De taxonomie is een veel gebruikte structuur voor het organiseren van categorieën (Collins & Quillian, 1969; McClelland & Rogers, 2003; Smith & Kosslyn, 2009). Het is een indeling van concepten in een set hiërarchisch gestructureerde categorieën, waarbij specifiekere, lagere orde, categorieën deel uitmaken van de meer globale, hogere orde, categorieën. Zie voor een taxonomie van objecten Figuur 1.



Figuur 1. Een taxonomie van objecten, waarbij meer algemene categorieën bestaan uit meer specifieke categorieën. Uit: *Cognitive psychologie: Mind and brain* (p.186), door Smith, E.E. & Kosslyn, S.M., 2009, New Jersey: Pearson Education.

Progressieve differentiatie van concepten houdt in dat men eerst leert globale categorieën van concepten te differentiëren alvorens men leert steeds specifiekere categorieën te differentiëren. Onderzoek van o.a. Mandler (2000) en Pauen (2002) leveren bewijs voor progressieve diffentiatie. Kinderen tussen de zeven en negen maanden oud bleken namelijk al in staat dieren en meubels van elkaar te onderscheiden. Objecten die minder van elkaar verschillen, zoals verschillende typen meubels, werden echter pas later correct van elkaar onderscheiden. Bij deze onderzoeken werd gekeken naar conceptuele categorisatie, waarbij objecten van elkaar worden onderscheiden op basis van hun functie. Er werd daarom gecorrigeerd voor perceptuele categorisatie, waarbij concepten worden gecategoriseerd op basis van hun perceptuele gelijkenis. Conceptuele categorisatie wordt gebruikt voor inductie in tegenstelling tot perceptuele categorisatie die wordt gebruikt voor herkenning.

Een ander bewijs voor de progressieve differentiatie van meer globale categorisatie naar meer specifieke categorisatie is het feit dat naarmate men meer ervaring op een bepaald gebied heeft, men steeds beter wordt in het verwerken categorieën van lagere orde (Gauthier,

Tarr, Anderson, Skudlarski, & Core, 1999). Experts kunnen categorieën van lagere orde zelfs even goed verwerken als de categorieën van hogere orde. Dit leidt tot progressieve differentiatie als eerste richtlijn voor het modelleren van categorisatie.

1.1.2. Prototypes

Een prototype van een categorie representeert een exemplaar uit deze categorie die alle meest voorkomende eigenschappen bezit (Smith & Kosslyn, 2009). Op deze manier is een prototype dus niets anders dan een verzameling van statistische gegevens over de meest voorkomende eigenschappen binnen een categorie. Leden van een categorie die veel eigenschappen met het prototype delen en die dus sterk op het prototype lijken worden "typisch" genoemd. Het "typisch" zijn van een exemplaar kan worden uitgedrukt in het aantal eigenschappen dat het exemplaar met het prototype deelt en is daarom een continue variabele.

Mervis & Pani (1980) toonden aan dat categorieën gemakkelijker werden geleerd na aanbieding van typische exemplaren van de categorie dan na aanbieding van minder typische exemplaren. Rosch (1975) toonde daarnaast aan dat typische leden van een categorie sneller gecategoriseerd worden. Posner & Keele (1968) voegden hier ten slotte aan toe dat typische leden ook met een grotere nauwkeurigheid worden gecategoriseerd dan niet typische leden van een categorie. Het voorgestelde model dient dus typische exemplaren van een categorie sneller en nauwkeuriger te categoriseren dan niet typische exemplaren.

1.1.3. Dynamische representatie

Dynamische representatie refereert naar de mogelijkheid om verschillende representaties van een categorie te construeren en op te roepen (Smith & Kosslyn, 2009). De representatie die het meest actief is, hangt af van de context. Dit kan worden geïllustreerd aan de hand van een voorbeeld. De categorie "zout" wordt in gedachten genomen. In de context van koken zal de eigenschap van zout dat die het eten extra smaak geeft vooral relevant zijn. Als men echter buiten in de sneeuw autorijdt, zal de eigenschap van zout dat die het het

vriespunt van water doet dalen, juist relevant zijn. Bij een dynamische representatie van het concept “zout” zal de eigenschap “smaakmaker” dus meer actief zijn bij het koken en de eigenschap “doet het vriespunt van water dalen” meer actief bij het autorijden in de sneeuw. Een model van categoriseren moet dus in staat zijn verschillende representaties van een bepaald concept op te roepen afhankelijk van de context.

1.2. Sequentieleren

Tijd is een essentieel begrip voor de mens. Zo krijgen de meeste handelingen slechts zin als ze in een bepaalde volgorde in de tijd worden uitgevoerd. Zo moeten bij het spelen van een lied op de piano bijvoorbeeld de toetsen in de juiste volgorde worden aangeslagen om de gewenste muziek te produceren. Een dergelijke volgorde wordt een sequentie genoemd. Google Dictionary (2012) definieert een sequentie als volgt: *“A particular order in which related events, movements, or things follow each other”.* Mensen leren in hun leven zeer veel sequenties. Denk naast het bovengenoemde voorbeeld aan het leren van routes, computerhandelingen of het bereiden van eten.

Ook bij sequentieleren doen zich fenomenen voor die een geschikt model moet kunnen produceren. De fenomenen die zullen worden besproken, zijn: het optreden *primacy* en *recency* effecten, het effect van sequentielengte op het terugroepen van sequenties en het omgaan met herhalingen in sequenties. Deze besprekingen worden wederom afgesloten met richtlijnen voor het model van sequentieleren.

1.2.1. *Primacy* en *recency* effect

Het *primacy* effect wordt door Gleitman, Gross, & Reisberg (2010) gedefinieerd als: *“in free recall, the tendency to recall the first items on the list more readily than those in the middle” (p.304)*. Het *recency* effect betreft de volgende definitie: *“in free recall, the tendency to recall items at the end of the list more readily than those in the middle” (p.304).* Dit komt erop neer dat elementen aan het begin of aan het einde van een bestudeerde sequentie vaker

correct worden opgeroepen uit het geheugen dan elementen in het midden van een sequentie. Verschillende onderzoeken hebben deze effecten, waaronden die van Deese & Kaufman (1957) en Murdock (1962). Bij *free recall* wordt geen rekening gehouden met de volgorde van de elementen in een sequentie. Echter, ook bij *serial recall*, waar de elementen ook in de juiste volgorde teruggeroepen dienen te worden, zijn *primacy* en *recency* effecten gevonden (Jahnke, 1965).

Het *primacy* effect wordt door Gleitman, Gross, & Reisberg (2010) verklaard door het toekennen van aandacht aan de elementen uit de sequentie. Bij het waarnemen van het eerste element van een sequentie kan de aandacht volledig op dit element worden gericht en herhaald de proefpersoon dit element in zichzelf. Bij het waarnemen van het tweede element moet de aandacht worden verdeeld over de twee elementen. Deze verdeling van aandacht over de verschillende elementen gaat door tot het einde van de sequentie is bereikt. Op deze wijze hebben de elementen aan het begin van de sequentie meer aandacht gekregen, waardoor ze hoogstwaarschijnlijk beter zijn verwerkt en in het lange termijn geheugen zijn opgeslagen. Deze verklaring wordt ondersteund door Marshall & Werder (1972) die aantoonden dat het *primacy* effect sterker was als er meer tijd tussen het aanbieden van de elementen zat, zodat er meer tijd was de elementen te herhalen.

Het *recency* effect wordt door Gleitman, Gross, & Reisberg (2010) verklaard door een beperkte capaciteit van het werkgeheugen. Het werkgeheugen wordt geacht 7 ± 2 elementen te kunnen vasthouden (Miller, 1956). Zodra een sequentie langer is, worden de eerdere elementen van de sequentie uit het werkgeheugen gestoten. Bij het oproepen van de sequentie worden de laatste elementen dus vaker correct opgeroepen, omdat ze nog in het werkgeheugen actief zijn. Bjork & Whitten (1974) hebben echter laten zien dat het *recency* effect ook plaatsvindt onder omstandigheden, waarbij men ervan uit gaat dat de laatste elementen ook al uit het werkgeheugen gestoten zijn. Ondanks het feit dat er over de

verklaring van het *primacy* en *recency* effect nog discussie is, valt over het bestaan van deze effecten niet te twisten. Het optreden van *primacy* en *recency* effecten is daarom belangrijk voor het modelleren van menselijke cognitie en vormt de eerste richtlijn om sequentieleren te modelleren.

1.2.2. Sequetielengte

Er bestaat een duidelijke relatie tussen het oproepen van een sequentie uit het geheugen en de lengte van de sequentie (Botvinick & Plaut, 2006; Lewandowsky & Murdock, 1989). De proportie correct opgeroepen elementen van een sequentie daalt namelijk wanneer de sequentie uit meer elementen bestaat. Deze relatie tussen accuratesse en sequentielengte volgt een sigmoïde patroon, oftewel heeft een *S*-vorn. Dit betekent dat verschillende hele korte sequenties met ongeveer dezelfde nauwkeurigheid worden opgeroepen. Dit geldt ook voor verschillende hele lange sequenties. Tussen middellange sequentie bevinden zich echter wel grotere verschillen in nauwkeurigheid van oproepen. Dit leidt tot de volgende richtlijn: het voorgestelde model moet om dezelfde invloed te laten zien van sequentielengte op het oproepen van de sequentie dus in ieder geval kortere sequenties met grotere nauwkeurigheid kunnen leren dan langere sequenties.

1.2.3. Herhalingen in sequenties

Het is mogelijk dat in sequenties bepaalde elementen worden herhaald. Om het leren van sequenties effectief te modelleren is het van belang te weten hoe het brein met deze herhalingen omgaat. Als men bijvoorbeeld alleen naar het vorige element kijkt om het volgende te voorspellen, kunnen herhalingen problemen opleveren. Immers, als op het ene punt element “C” wordt gevolgd door element “D” en op een ander punt door element “E” kan men bij het oproepen van de sequentie niet weten welk element na “C” komt. Om te weten hoe het brein met herhalingen in sequenties omgaat, wordt daarom gekeken hoe het brein sequenties representeert.

Henson (1998) beschrijft drie theorieën die de representatie van sequenties in het brein beschrijven, namelijk: *chaining theory*, *ordinal theory* en *positional theory*. Iedere theorie heeft sterke en zwakke punten. De literatuur geeft daarom ook nog geen eenduidig beeld over welke theorie de voorkeur krijgt. Hieronder zullen de theorieën besproken worden aan de hand van hun implicaties betreffende het optreden van herhalingen in sequenties.

De *chaining theory* beschrijft de representatie van sequenties in termen van associaties die worden gelegd tussen de verschillende items in een sequentie (Henson, 1998; Schuck, Gaschler, Keisler, & Frensch, 2012; Smith & Kosslyn, 2009). Dit wordt ook wel item-item associatie genoemd. De sterkste associatie wordt gelegd tussen een bepaald item en het item dat daarop volgt. Op deze manier vormt ieder item uit de sequentie een aanwijzing (*cue*) voor het oproepen van het volgende item uit het geheugen. Dit leidt tot de vraag hoe deze theorie omgaat met herhalingen van elementen in een sequentie, omdat de twee verschillende elementen die volgen dus door dezelfde aanwijzing moeten worden opgeroepen. Dit probleem wordt tot bepaalde hoogte opgelost door *compound chaining* (Henson, 1998), waarbij de aanwijzing niet slechts uit één element bestaat, maar uit meerdere elementen die aan het op te roepen element voorafgaan. Dit verkleint de ambiguïteit van de aanwijzing. Het *chaining* principe kan verklaren dat veel mensen bij het noemen van het laatste cijfer van hun bankrekeningnummer eerst het hele nummer afgaan (Smith & Kosslyn, 2009).

De *ordinal theory* gaat uit van de veronderstelling dat de volgorde van elementen in een sequentie wordt bepaald door de relatieve afstand tussen de elementen in plaats van de absolute afstand (Henson, 1998). Page & Norris (1998) beschrijven het *primacy model* waarin deze theorie wordt toegepast. Hierin wordt beschreven dat elementen vroeg in een sequentie steeds meer geactiveerd zijn dan elementen later in de sequentie. Bij het oproepen van de sequentie uit het geheugen wordt eerst het item met de grootste activatie opgeroepen. Deze wordt vervolgens onderdrukt waarna het volgende element met de dan grootste activatie

wordt opgeroepen. Zo worden alle elementen in de juiste volgorde opgeroepen uit het geheugen. Bij dit model moet de veronderstelling van een token representatie van de elementen worden aangenomen om problemen met herhalingen in sequenties te voorkomen (Henson, 1998). Op deze manier kunnen de herhaalde elementen namelijk toch een verschillende activatiewaarde hebben wat niet het geval is bij een type representatie. Types en tokens verschillen van elkaar in de zin dat een type een bepaald concept is in tegenstelling tot een token wat een exemplaar van een concept is. Een type kan dus verschillende tokens hebben. Dit wordt geïllustreerd in het volgende voorbeeld: het type "hoofdletter A" is abstract en niet fysiek, maar bestaat uit verschillende fysieke tokens, zoals "A", "A", "𝒜" of "**A**". De herhaalde elementen worden in het *primacy model* dus beschouwd als exemplaren (die verschillende eigenschappen kunnen hebben) van hetzelfde type.

De *positional theory* ten slotte beschrijft de representatie van sequenties in termen van associaties die gelegd worden tussen de items uit de sequentie en de positie die ze innemen, oftewel item-positie associaties (Henson, 1998; Schuck et al., 2012). Volgens deze theorie krijgt ieder element uit een sequentie dus als het ware een label met zijn positie toegekend wat Smith, & Kosslyn (2009) *"order tags"* noemen. Deze theorie heeft dus geen problemen met herhalingen in een sequentie, omdat de herhalingen ieder een eigen positie hebben. De theorie kent echter wel andere problemen (Smith & Kosslyn, 2009; Schuck et al., 2012).

Als deze theorieën in acht worden genomen, kan geconcludeerd worden dat er rekening moet worden gehouden met de representatie van sequenties om op een adequate manier met herhalingen in een sequentie om te gaan. Er moet een representatie worden gebruikt, waarbij herhalingen geen problemen vormen of waarbij maatregelen worden genomen om problemen met herhalingen te voorkomen.

1.3. Modelleren van cognitie

Er zijn verschillende benaderingen om cognitie te modelleren, zoals de symbolische benadering (Dinsmore, 1992), de gestructureerde probabilistische benadering (McClelland, Botvinick, Noelle, Plaut, Rogers, Seidenberg, & Smith, 2010) en het connectionisme (Bechtel & Abrahamsen, 2002; McClelland et al., 2010). De symbolische benadering ziet het brein als het ware als een computer, waarbij cognitie niets anders is dan het manipuleren van symbolen om een bepaald doel te bereiken (Dinsmore, 1992). Deze symbolen kunnen representaties zijn van de buitenwereld en kunnen zo ook betekenis hebben. Gestructureerde probabilistische modellen worden door McClelland et al. (2010) beschreven als: *"models that specify that cognitive activity involves the use of probabilistic information to select among and specify the parameters of particular structural forms that specify the relationships among items represented by discrete symbols" (p.348).* Het connectionisme ten slotte is gebaseerd op het idee dat cognitieve activiteit ligt opgeslagen in de verbindingen tussen neuronen. Kennis wordt vervolgens verkregen door de sterkte van deze verbindingen aan te passen op basis van ervaring (McClelland et al., 2010).

Een belangrijk sterk punt van connectionistische modellen is dat ze rekening houden met de mechanismen die aan cognitie ten grondslag liggen in tegenstelling tot de andere twee benaderingen die zich vooral richten op het doel van cognitie (McClelland et al., 2010). Dit punt uit zich in het volgende voorbeeld beschreven door McClelland et al. (2010). Het vermogen om te zien is niet alleen ontwikkeld vanuit het probleem dat een organisme niet in staat is te kunnen zien. De ontwikkeling van het vermogen om te zien is ook beperkt door evolutiemogelijkheden. Bij het modelleren van cognitieve processen moet dus ook rekening gehouden worden met biologische grenzen die het aantal mogelijke onderliggende mechanismen beperken.

Een ander sterk punt van het connectionisme is dat het eigenschappen heeft die overeen komen met eigenschappen van het brein. Zo vertonen connectionistische modellen *graceful degradation*, een proces waarbij beschadiging van een aantal modelcomponenten geen al te grote gevolgen heeft voor de functie van het model (Purves et al., 2008). Dit komt overeen met de werking van het brein. Het *mass action principle* stelt namelijk dat de mate waarin het vermogen te leren is aangetast zich verhoudt tot de hoeveelheid beschadigd hersenweefsel (Purves et al., 2008). Om deze twee redenen wordt in dit onderzoek een connectionistische benadering wordt gebruikt om categoriseren en sequentieleren te modelleren.

De connectionistische benadering maakt gebruik van neurale netwerken om cognitie te modelleren (Bechtel & Abrahamsen, 2002). Een neuraal netwerk is een dynamisch systeem dat bestaat uit eenheden met een bepaalde activatiewaarde. De eenheden staan met elkaar in verbinding en kunnen elkaar exciteren of inhiberen als het systeem van een bepaalde input wordt voorzien, net zoals neuronen in de hersenen dat kunnen. De eenheden in een neuraal netwerk worden daarom ook wel neuronen genoemd. Er zijn verschillende manieren waarop neurale netwerken vormgegeven kunnen zijn. De architectuur van een netwerk bepaalt welke problemen te leren of uit te voeren zijn. Hieronder zullen twee netwerkarchitecturen worden besproken die de eerder genoemde richtlijnen grotendeels volgen en die gemakkelijk zijn te integreren met elkaar. Dit zijn het *Feedforward Network* van Rumelhart als model van categorisatie (McClelland & Rogers, 2003; Rogers & McClelland, 2008) en het *Simple Recurrent Network* van Elman als model van sequentieleren (Elman, 1991).

1.3.1. Feedforward model van Rumelhart

Het neurale netwerk van Rumelhart (McClelland & Rogers, 2003; Rogers & McClelland, 2008) is in dit onderzoek gebruikt om categorisatie te modelleren. Dit is een speciaal type *feedforward network*. Een *feedforward network* bestaat uit verschillende lagen

neuronen: een laag inputneuronen, een laag outputneuronen en meestal een of meer lagen verborgen neuronen (Bechtel & Abrahamsen, 2002). Het netwerk genereert zelfstandig output door de inputeenheden een bepaalde activatiewaarde mee te geven. Deze activatie verspreidt zich vervolgens door het netwerk via de connecties tussen de neuronen. De activatiestroom gaat slechts in één richting, van de inputneuronen via de verborgen neuronen naar de outputneuronen.

De activatiewaarde van een willekeurig neuron "x" wordt bepaald aan de hand van Vergelijking 1. Het eerste gedeelte van deze functie is het effect van de huidige activatie van neuron "x" op de nieuwe activatie (Van der Velde, 2011a). Hoe groot dit effect is, hangt af van α, een factor die bepaalt in welke mate de oude activatie wordt behouden. In het geval dat α de waarde 0 aanneemt, heeft de oude activatie geen invloed op de nieuwe activatiewaarde van een neuron.

$$Act_x = Act_{x.oud} \cdot \alpha \; + \; \frac{1}{1+e^{-p \cdot (\sum(Act_i \cdot W_i) - \beta)}} \; . \qquad (1)$$

Het tweede gedeelte van Vergelijking 1 is de activatiefunctie van het neuron. Aan de input van deze functie is te zien dat de activatiewaarde van neuron "x" wordt bepaald door de som te nemen van de activiatiewaarden van neuronen die "x" input geven (Act_i), vermenigvuldigd met hun connectiegewicht (W_i). Het connectiegewicht bepaalt of het neuron geïnhibeerd of geëxciteerd wordt en in welke mate. Daarnaast wordt er rekening gehouden met de weerstand die een bepaald neuron biedt tegen het veranderen van zijn actievatiewaarde door van deze som een *treshhold* (β) af te trekken. De logistische activatiefunctie die hier is gebruikt, zorgt er vervolgens voor dat het grote bereik aan mogelijke inputwaarden wordt gereduceerd tot een bereik van 0 tot 1 voor de outputwaarden. De helling van de functie kan aangepast worden aan de hand van factor p. Er is dus geen één-op-éénrelatie tussen de input van een neuron en zijn output.

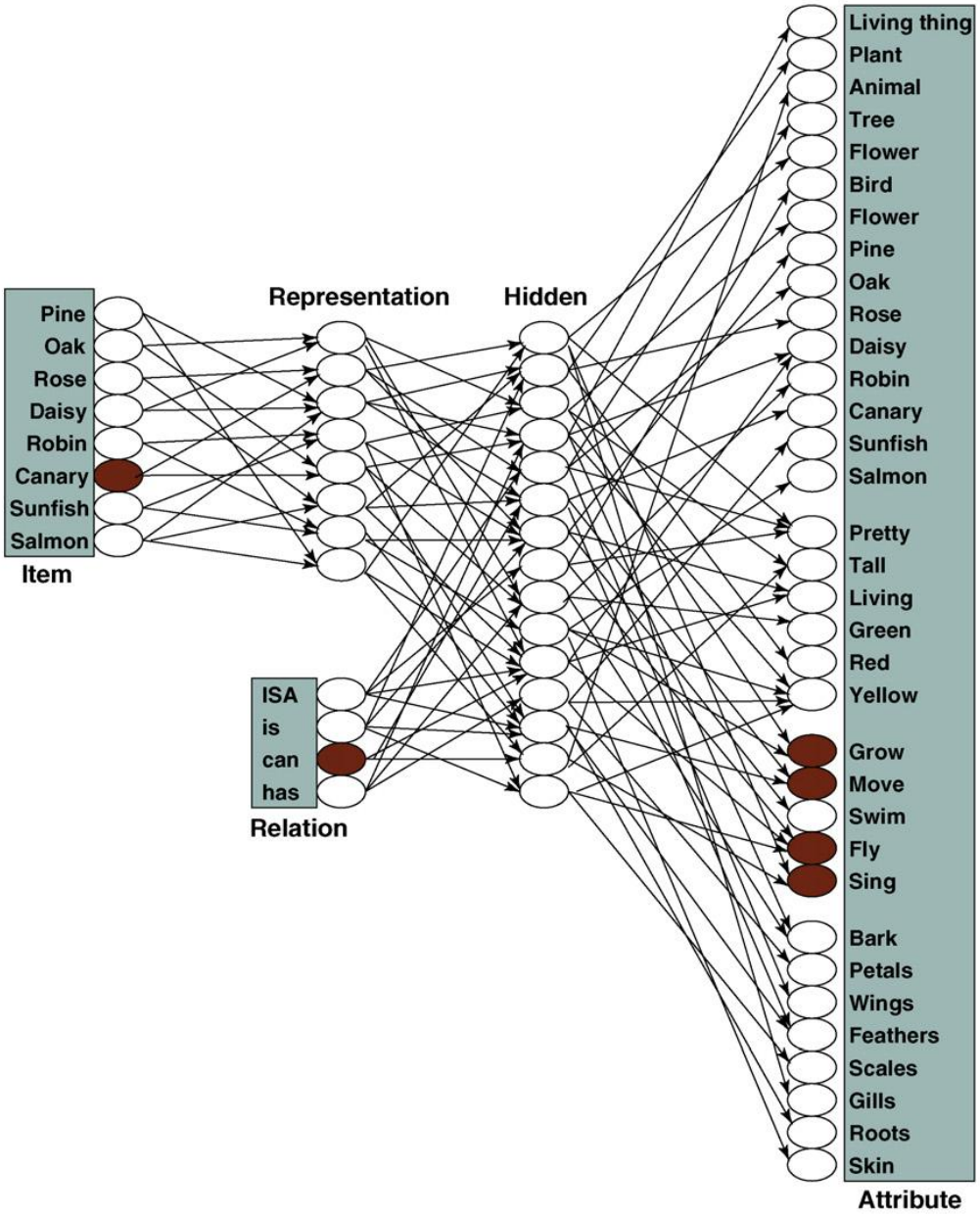
De spreiding van activatie door het netwerk leidt niet direct tot de gewenste output. Het netwerk moet eerst "leren" om de gewenste output te genereren door de connectiegewichten aan te passen. Dit leerproces kan plaatsvinden door gebruik te maken van *backpropagation* (Hecht-Nielsen, 1989). Hoe dit leerproces precies plaatsvindt, zal in de methode worden uitgelegd.

Het *feedforward network* van Rumelhart is in staat semantische relaties te leren en weer te geven (Rogers & McClelland, 2008). In Figuur 2 is een implementatie van het Rumelhart netwerk weergegeven, waarin te zien is hoe semantische relaties geleerd kunnen worden (Rogers & McClelland, 2008). Er wordt eerst bepaalde input (in dit voorbeeld "*canary"*) en context (in dit voorbeeld *"can"*) aan het netwerk aangeboden. Vervolgens genereert het netwerk een output door de activatiestroom die de input veroorzaakt. Na het leren van het netwerk, is het netwerk in staat de gewenste output te genereren (in dit voorbeeld *"grow", "move", "fly"* en *"sing"*). Het netwerk is zo dus in staat weer te geven wat een kanarie allemaal kan. Dit gaat op analoge wijze voor de andere concepten. Uiteindelijk is het netwerk dan in staat om van alle concepten de eigenschappen weer te geven, bijvoorbeeld: "een kanarie kan groeien" of "een kanarie is een vogel".

Daarnaast is het netwerk ook in staat om de concepten te categoriseren op basis van hun eigenschappen. Bij activatie van de eigenschap "is een vogel", worden bijvoorbeeld de concepten "kanarie" en "roodborstje" actief. Nadere analyse van de verborgen neuronen in het netwerk toont aan dat deze categorisatie inderdaad plaatsvindt via progressieve differentiatie en dat typische exemplaren van een categorie inderdaad eerder geleerd worden (Rogers & McClelland, 2008).

Wat het netwerk van Rumelhart een bijzonder *feedforward network* maakt, is het feit dat dit netwerk context kan representeren. Dit zorgt ervoor dat het netwerk verschillende situaties kan onderscheiden. Zo kan het weergegeven netwerk het onderscheid leren tussen

"een kanarie kan groeien" en "een kanarie heeft veren". Bij dezelfde input ("kanarie") weet het netwerk door de verschillende contexten ("kan" en "heeft") welke output gewenst is ("groeien" of "veren"). Het netwerk is dus in staat de concepten dynamisch te representeren.



Figuur 2. Het *feedforward network* van Rumelhart dat semantisch geheugen modelleert: Uit "Précis of semantic cognition: A parallel distributed processing approach" door Rogers, T.T. & McClelland, J.L., 2008, *Behavioral and Brain Sciences, 31*, p.692.

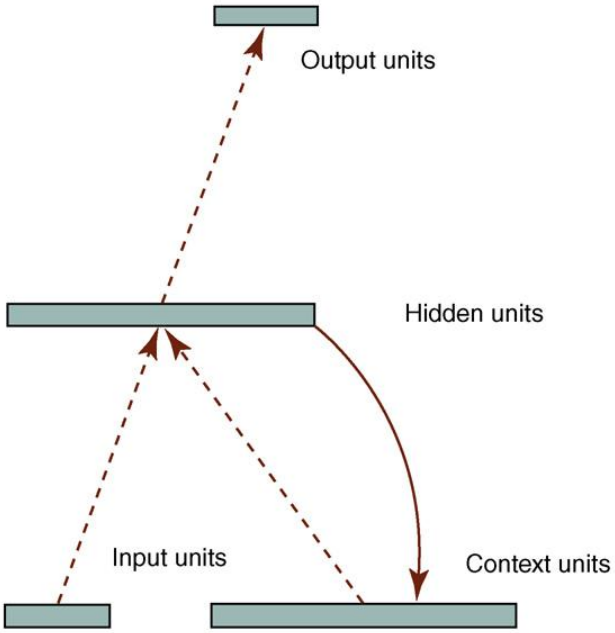
1.3.2. Simple recurrent network van Elman

Het netwerk van Rumelhart is niet geschikt om het leren van sequenties te modelleren, omdat het geen volgorde kan representeren. Het netwerk is niet in staat te weten wat de vorige

stap of de vorige input van het netwerk was. Het mist dus in principe een soort van geheugen. Een netwerk dat wel over deze capaciteit beschikt is het *simple recurrent network* van Elman, weergegeven in Figuur 3 (Elman, 1991).

Het *simple recurrent network* van Elman kan sequenties leren door gebruik te maken van *compound chaining* (Henson, 1998). Na aanbieding van het eerste item van een sequentie, wordt dit item in het netwerk gerepresenteerd door de activatiewaarden van de verborgen neuronen. Bij aanbieding van het tweede item, wordt ook de representatie van het eerste item aan het netwerk aangeboden. Door deze gekoppelde aanbieding worden het eerste en het tweede item van de sequentie aan elkaar geassocieerd. In de figuur gebeurt dit via de contextneuronen. Bij aanbieding van het derde item krijgt het netwerk via de contextneuronen ook de status van het netwerk na aanbieding van het tweede item aangeboden. Deze status is een combinatie van de representaties van het eerste en van het tweede item. Een bepaald item uit een sequentie wordt dus samen aangeboden en geassocieerd met een representatie van alle vorige items uit de sequentie. Door het netwerk te trainen, is het in staat, bij een bepaalde input, het volgende item uit de sequentie te voorspellen.

Zoals eerder is genoemd, lost *compound chaining* het probleem van herhalingen in een sequentie tot op zekere hoogte op (Henson, 1998). Het *simple recurrent network* volgt dus ook tot op zekere hoogte de richtlijnen voor het modelleren van sequentieleren.



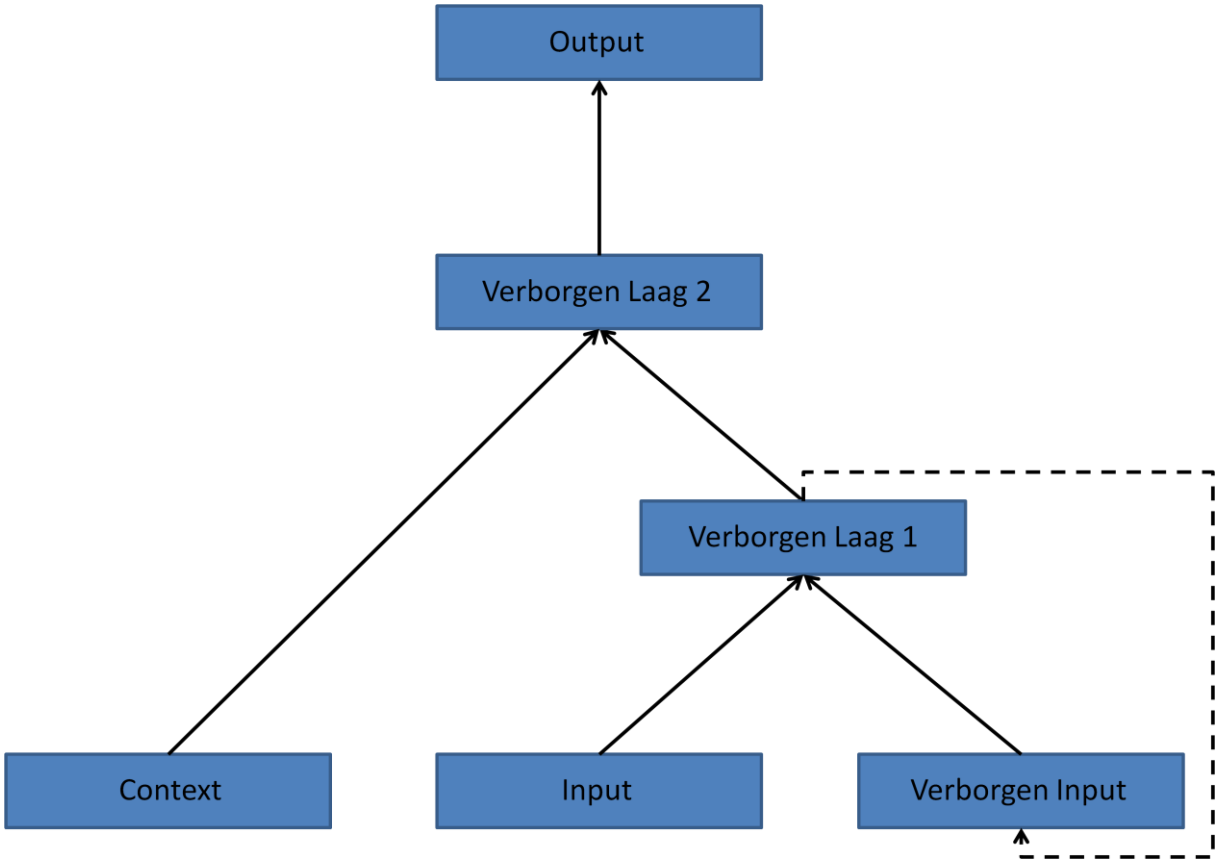
Figuur 3. Het *simple recurrent network* van Elman: Uit "Distributed representations, simple recurrent networks, and grammatical structure" door Elman, J.L., 1991, *Machine Learning, 7,* p.200.

1.4. Categoriseren van sequenties

Het neurale netwerk van Rumelhart blijkt dus in staat concepten op basis van bepaalde eigenschappen en context te categoriseren. Het *simple recurrent network* van Elman is in staat sequenties te leren. Deze netwerken zijn geschikt om te integreren, omdat beiden netwerken een type *feedforward network* zijn (Bechtel & Abrahamsen, 2002) en beiden leren met behulp van *backpropagation* (Elman, 1991; McClelland & Rogers, 2003). Ondanks het feit dat het *simple recurrent network* van Elman een recurrent netwerk wordt genoemd, kan het ook als een *feedforward network* beschouwd worden, omdat de recurrente connecties alleen worden gebruikt om het netwerk van input te voorzien. Bij het genereren van output worden echter alleen de voorwaartse connecties tussen neuronen gebruikt. De vraag is nu of een combinatie van deze twee netwerken nog steeds sequenties kan leren en nog steeds kan categoriseren; oftewel of het in staat is sequenties te categoriseren.

Het is van belang om na te gaan of een combinatie van twee werkende modellen nog steeds naar behoeve functioneert. Ballard & Sprague (2006) stellen namelijk dat menselijk gedrag is opgebouwd uit microgedrag. Microgedragingen vormen dus de bouwstenen voor macrogedrag. De modellen van Rumelhart en Elman simuleren zulk microgedrag, namelijk categoriseren en sequentieleren. Om een representatie van de werkelijkheid te geven, zouden deze modellen dus ook bouwstenen moeten zijn voor een model dat macrogedrag, namelijk het categoriseren van sequenties, modelleert.

Om sequenties te categoriseren, moet het model eerst in staat zijn sequenties te discrimineren alvorens ze op basis van gemeenschappelijke eigenschappen te kunnen categoriseren. Het *simple recurrent network* van Elman geeft echter geen context aan het netwerk mee en kan in tegenstelling tot het netwerk van Rumelhart dus niet discrimineren tussen situaties. Dit betekent ook dat het netwerk in zijn huidige staat niet kan discrimineren tussen overlappende sequenties. Het neurale netwerk dat wordt voorgesteld als model van het categoriseren van sequenties is daarom de combinatie van het netwerk van Rumelhart en het *simple recurrent network* van Elman die staat weergegeven in Figuur 4. Bruijnes (2011) toonde al aan dat dit netwerk in staat is verschillende sequenties te discrimineren. Nu wordt gekeken of de sequenties ook gecategoriseerd kunnen worden.



Figuur 4. Het voorgestelde model om sequenties te categoriseren: een integratie van het *feedforward network* van Rumelhart en het *simple recurrent network* van Elman. De recurrente verbindingen zijn met de onderbroken pijl weergegeven.

Om na te gaan of het voorgestelde model succesvol is in het categoriseren van sequenties, zijn de volgende hypotheses opgesteld:

Hypothese 1. Het voorgestelde model is in staat sequenties te leren.

Hypothese 2. Het voorgestelde model is in staat op basis van een context de geleerde sequenties te categoriseren.

Bij het toetsen van deze hypotheses zullen de richtlijnen uit de inleiding in acht worden genomen. De richtlijnen voor het categoriseren en het sequentieleren staan daarom hieronder nogmaals weergegeven.

Richtlijnen voor categoriseren zijn:

1. Het model categoriseert via het proces van progressieve differentiatie, waarbij globale categorieën eerder geleerd worden dan specifieke categorieën.
2. Het model categoriseert “typische” exemplaren van een categorie sneller en met grotere nauwkeurigheid dan niet “typische” exemplaren.
3. Het model is in staat verschillende representaties van bepaald concept op te roepen afhankelijk van de context.

Richtlijnen voor sequentieleren zijn:

1. Het model toont *primacy* en *recency* effecten bij het leren van sequenties, wat betekent dat elementen aan het begin en aan het einde van de sequentie met grotere nauwkeurigheid worden geleerd dan elementen in het midden.
2. Het model leert kortere sequenties met grotere nauwkeurigheid dan langere sequenties.
3. Het model moet in staat zijn om te gaan met het voorkomen van herhaalde elementen in een sequentie zonder dat het leren van de sequenties wordt beïnvloed.

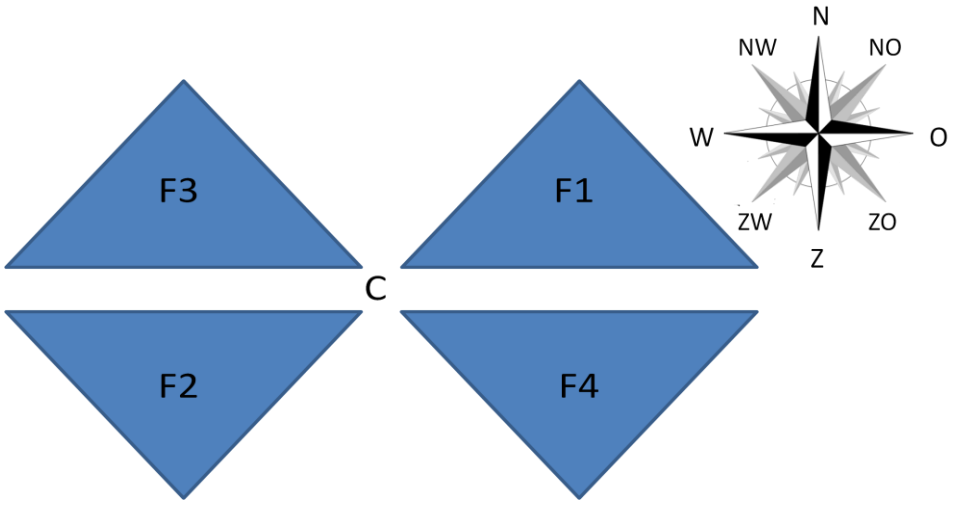
Om het voorgestelde model van het categoriseren van sequenties te testen en de hypotheses te toetsen, zijn verschillende simulaties uitgevoerd. In het volgende hoofdstuk zal eerst de algemene methode worden besproken die voor alle simulaties geldt. Vervolgens zullen de simulaties apart besproken worden, waarbij toevoegingen aan de algemene methode, de resultaten en de discussie aan bod komen. Het onderzoek wordt tenslotte afgesloten met een algemene discussie.

2. Algemene Methode

Er is nagegaan of het voorgestelde neurale netwerk geschikt is om sequenties te categoriseren. Hieronder zal de gebruikte methode worden besproken. Eerst zal de taak besproken worden die het netwerk heeft moeten uitvoeren. Vervolgens zal worden besproken hoe het netwerk is gemodelleerd, getraind en getest. Ten slotte wordt besproken hoe het netwerk geanalyseerd zal worden. De resultaten van de analyse komen in het volgende hoofdstuk aan bod.

2.1. Taak: routes leren

Om na te gaan of het neurale netwerk geschikt is om sequenties te categoriseren, is een taak bedacht waarmee dit getest kan worden. De taak bestaat uit het categoriseren van routes die gemaakt zijn aan de hand van de vier figuren uit Figuur 5. Iedere route begint en eindigt in hetzelfde punt “C” en wordt gevormd door de omtrek van een figuur te volgen. De omtrek van ieder figuur kan in twee richtingen worden doorlopen, afhankelijk van de draairichting die wordt gekozen (linksom of rechtsom). Dit leidt tot een totaal van 8 routes die worden geleerd.



Figuur 5. Figuren waarnaar de sequenties (routes) gemaakt zijn. De routes worden gevormd door vanaf startpunt “C” de omtrek van de figuren te volgen in beide richtingen.

De routes worden uitgedrukt in de windrichtingen die gevolgd moeten worden om van de ene hoek van het figuur naar de volgende hoek te komen totdat de gehele figuur is doorlopen. Zo ontstaan sequenties die de verschillende routes representeren. De route waarbij F1 linksom wordt doorlopen, wordt bijvoorbeeld volgens deze sequentie genoteerd: C – oost – noordwest – zuidwest – C. Vanaf het startpunt "C" moet men dus in oostelijke richting gaan om het punt rechtsonder te bereiken. Daar aangekomen moet men in noordwestelijke richting gaan om het punt bovenaan de figuur te bereiken. Vanuit dat punt gaat men in zuidwestelijke richting om ten slotte eindpunt "C" te bereiken. Op analoge wijze zijn de overige zeven sequenties geformuleerd, zie Tabel 1.

Tabel 1

Sequenties (routes) die het netwerk dient te leren samen met het figuurnummer en draairichting als context.

Figuur	Draairichting	Sequentie
F1	Linksom	C – oost – noordwest – zuidwest – C
F1	Rechtsom	C – noordoost – zuidoost – west – C
F2	Linksom	C – west – zuidoost – noordoost – C
F2	Rechtsom	C – zuidwest – noordwest – oost – C
F3	Linksom	C – noordwest – zuidwest – oost – C
F3	Rechtsom	C – west – noordoost – zuidoost – C
F4	Linksom	C – zuidoost – noordoost – west – C
F4	Rechtsom	C – oost – zuidwest – noordwest – C

De figuren zijn zo samengesteld dat keuze van draairichting niet al direct volgt uit de eerste stap. De draairichting linksom kan dus voorkomen bij oost, maar ook bij west als eerste stap. Ook andere variabelen zijn (zoveel mogelijk) gebalanceerd. Bijvoorbeeld komen niet alleen oost en west, maar ook andere windrichtingen als eerste stap voor.

De taak van het neurale netwerk begint met het leren van de sequenties uit Tabel 1. Daarbij moet het netwerk in staat zijn de verschillende sequenties van elkaar te onderscheiden. Het figuurnummer en de draairichting vormen samen de context die het netwerk nodig heeft om hiertoe in staat te zijn. Vervolgens moet het netwerk de sequenties categoriseren naar draairichting.

2.2. Netwerkarchitectuur

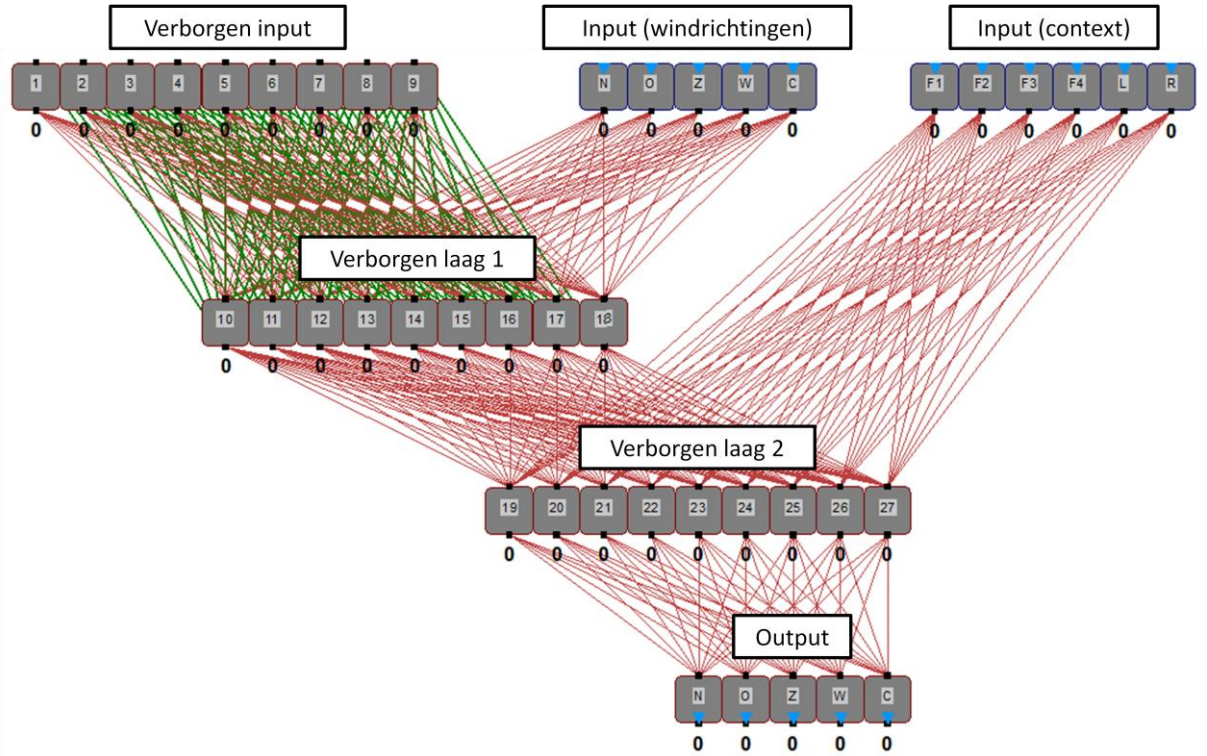
Het voorgestelde neurale netwerk is gemodelleerd met behulp van het computerprogramma *MemBrain Neurale Netze Editor und Simulator* (Membrain, 2010). Met dit programma kunnen neurale netwerken gemaakt, getraind en getest worden. Het voordeel van dit programma tegenover programma's als MATLAB 8 (MathWorks, 2012) is het gebruik van een grafische gebruikersinterface. Dit zorgt voor eenvoudig gebruik en maakt kennis van programmeren overbodig.

De architectuur, zoals in MemBrain weergegeven, van het voorgestelde netwerk is te zien in Figuur 6. De input bestaat uit twee onderdelen. Het eerste deel bestaat uit neuronen die alle elementen uit de verschillende sequenties representeren. Dit zijn dus de vier windrichtingen en het punt "C" wat leidt tot een totaal van vijf neuronen. Noordwestelijke richting, bijvoorbeeld, wordt in het netwerk aangeduid door de neuronen die "noord" en "west" representeren te activeren. Het tweede deel van de input bestaat uit neuronen die de context van de sequenties representeren. Dit zijn de vier figuurnummers en de twee draairichtingen wat leidt tot een totaal van zes neuronen.

De verborgen neuronen zijn verdeeld over drie lagen. Het aantal verborgen neuronen dat nodig is voor een optimale prestatie is niet bekend. Het aantal verborgen neuronen in dit netwerk is bepaald door een informele analyse. Hierbij is de vuistregel aangehouden dat het maximale aantal verborgen neuronen per laag gelijk is aan het aantal input neuronen. De eerste laag verborgen neuronen is te vinden naast de overige input. Dat komt doordat de eerste

laag verborgen neuronen input geeft aan de tweede laag verborgen neuronen. Deze input geeft informatie over het vorige element uit de sequentie dat aan het netwerk werd aangeboden. De output ten slotte bestaat uit één laag van vijf outputneuronen. Deze representeren wederom de mogelijke elementen uit de sequenties, namelijk de vier windrichtingen en het punt “C”.

De verbindingen tussen alle lagen zijn weergegeven in Figuur 6. Hierin zijn voorwaartse en recurrente netwerkverbindingen te onderscheiden. Bij het bekijken van de figuur moet rekening worden gehouden met het feit dat bij een verbinding tussen twee lagen alle neuronen uit de ene laag verbonden zijn met alle neuronen uit de tweede laag.



Figuur 6. Het neurale netwerk voor het categoriseren van sequenties, zoals het in MemBrain wordt weergegeven. De recurrente verbindingen zijn te herkennen aan hun groene kleur.

2.3. Leren van de sequenties

De spreiding van activatie door het netwerk als gevolg van het aanbieden van bepaalde input leidt niet direct tot de gewenste output. Een wijze om het netwerk te leren de gewenste output te genereren is door gebruik te maken van *backpropagation* (Bechtel & Abrahamsen, 2002; Hecht-Nielsen, 1989). Bij deze methode wordt gebruik gemaakt van het verschil tussen de door het netwerk gegenereerde output en de gewenste output. Dit verschil, de error, wordt

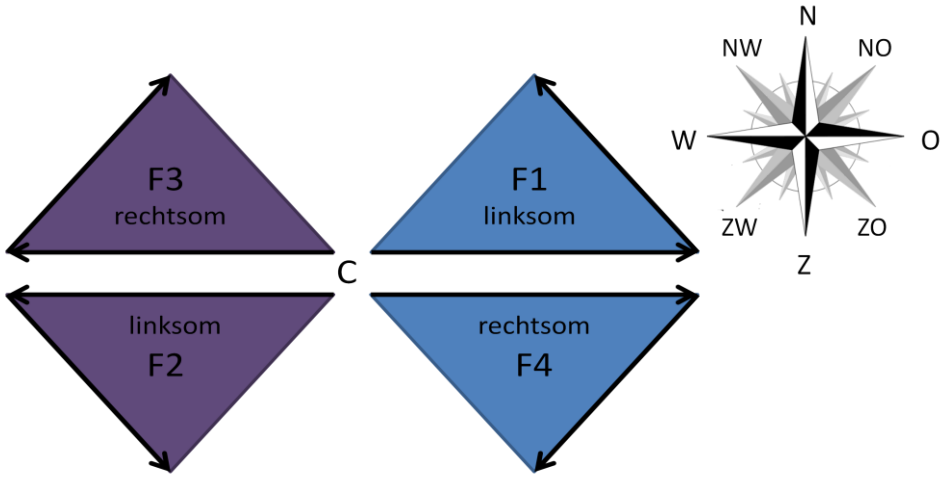
vervolgens verkleind door het toepassen van *backpropagation*. Hierbij wordt eerst nagegaan in welke mate alle mogelijke connecties tussen neuronen bijdragen aan het plaatsvinden van de error (Van der Velde, 2011b). Vervolgens wordt de bijdrage aan de error verminderd door de connectiegewichten aan te passen. Dit proces herhaalt zich totdat het netwerk nagenoeg de gewenste output genereert.

Voor het neurale netwerk uit dit onderzoek ziet de trainingsfase er als volgt uit. Eerst worden de connectiegewichten tussen de neuronen gerandomiseerd. Vervolgens wordt het eerste element van een sequentie met de daarbij horende context aangeboden aan het netwerk. Dit leidt tot een verspreiding van activatie via de connectiegewichten en resulteert in een bepaalde output. Deze output wordt vergeleken met de gewenste output en vervolgens wordt het verschil tussen verkregen en gewenste output verkleind door het toepassen van *backpropagation*. Hierbij moet rekening worden gehouden dat alleen de voorwaartse connecties worden aangepast en niet de recurrente connecties. Dit komt doordat de recurrente connecties slechts als doel hebben een kopie van de activatie van een laag verborgen neuronen door te geven aan een andere laag verborgen neuronen, zodat verschillende elementen uit een sequentie aan elkaar geassocieerd kunnen worden. Dit proces van het aanpassen van de voorwaartse connectiegewichten herhaalt zich voor ieder element uit een sequentie en voor alle acht sequenties uit Tabel 1. Vervolgens wordt het gehele proces herhaald totdat een acceptabele error bereikt is. Hiermee wordt de trainingsfase afgesloten.

2.4. Analyse van het neurale netwerk

Eerst wordt er nagegaan of de gewenste sequenties inderdaad correct aan het netwerk zijn geleerd. Dit wordt gedaan door aan het netwerk nogmaals de sequenties aan te bieden terwijl de connectiegewichten constant worden gehouden. Het percentage correct gegenereerde output kan dan worden bepaald.

Vervolgens valt af te leiden of het neurale netwerk de sequenties ook kan categoriseren naar draairichting. Dit gebeurt aan de hand van situaties, waarbij twee sequenties differentiëren als gevolg van de draairichting. Deze differentiatie vindt plaats als een route linksom en een route rechtsom, na het startpunt “C” eerst dezelfde richting op gaan, waarna ze bij de tweede stap ieder een andere richting op gaan. De twee verschillende richtingen vormen eigenschappen van het linksom dan wel rechtsom draaien. Zo wordt er bijvoorbeeld bij F1 en bij F4 vanuit startpunt “C” naar oostelijke richting gegaan. Bij F1 is de volgende stap echter in noordwestelijke richting, vanwege een draairichting linksom. Bij F4 is de volgende stap juist in zuidwestelijke richting vanwege een draairichting rechtsom. Het netwerk zou dus moeten herkennen dat als het tweede element van de sequentie, “oost”, bij de categorie “linksom” hoort, de categorie de eigenschap heeft dat het tweede element wordt gevolgd door het item “noordwest” en niet “zuidwest”. In Figuur 7 zijn de vier sequenties waarbij een dergelijke differentiatie plaatsvindt te vinden. Dit zijn: de route van F1 met een draairichting linksom in combinatie met de route van F4 met draairichting rechtsom; en de route van F2 met draairichting linksom in combinatie met de route van F3 met draairichting rechtsom.



Figuur 7. De situaties waarbij het netwerk in staat moet zijn de sequenties te categoriseren naar draairichting. In de paarse combinatie van figuren dient het netwerk de routes die starten in westelijke richting te onderscheiden op basis van de draairichting. In de blauwe combinatie van figuren dient het netwerk de routes die starten in oostelijke richting te onderscheiden op basis van de draairichting.

Het netwerk moet dus “kennis” hebben van de eigenschappen die bij de categorieën linksom en rechtsom horen. Om na te gaan of dit het geval is, wordt aan het getrainde netwerk een nieuwe inputset aangeboden met slechts de draairichting als context en een representatie van het tweede item van de sequentie als input. Vervolgens wordt gekeken of het netwerk de vier sequenties uit de vorige alinea van elkaar kan onderscheiden. Dit wordt gedaan door naar de activatiewaarden van de outputneuronen te kijken. Een eerste stap in oostelijke richting en een draairichting linksom onderscheidt zich van een draairichting rechtsom door een tweede stap in noordelijke richting in plaats van zuidelijke richting. Dit betekent dat bij een draairichting linksom de activatiewaarde van het outputneuron “noord” groter moet zijn dan de activatie van “zuid”. Bij een draairichting rechtsom is dit precies andersom. Bij een eerste stap in westelijke richting, volgt bij een draairichting linksom een tweede stap in zuidelijke richting, in tegenstelling tot een tweede stap in noordelijke richting bij een draairichting rechtsom. Dit betekent dat bij “west” als eerste stap en een draairichting linksom, de activatiewaarde van het outputneuron “zuid” groter moet zijn dan de activatie van “noord”. Wederom is dit bij een draairichting rechtsom juist andersom.

Tenslotte kan worden nagegaan in hoeverre het netwerk *graceful degradation* vertoont. Beschadiging van een aantal modelcomponenten zou volgens dit proces geen al te grote gevolgen moeten hebben voor de functie van het model (Purves et al., 2008). In dit onderzoek kan daarom worden onderzocht hoe goed het netwerk is in het reproduceren van sequenties als slechts een gedeelte van de input wordt aangeboden die tijdens de trainingsfase werd aangeboden. Dit kan worden gedaan door wederom slechts de draairichting als context en een representatie van het tweede item van de sequentie als input te geven. Nu wordt gekeken of het netwerk inderdaad het derde element van de sequentie als output genereert. Dit wordt gedaan door te kijken of de outputactivatie van “noord” desgewenst hoger of lager is dan de outputactivatie van “zuid” én of de outputactivatie van “oost” desgewenst hoger of

lager is dan de outputactivatie van “west”. Is bijvoorbeeld het element “noordwest” het gewenste derde element uit een sequentie, dan dient de activatie van het outputneuron “noord” hoger te zijn dan de activatie van “zuid” én moet de activatie van het outputneuron “west” groter zijn dan de activatie van “oost”.

Bij de analyse wordt steeds nagegaan of de verwachte output wordt gegenereerd. Omdat dit een exploratief onderzoek is, zal geen verdere analyse worden gedaan naar de activatie van de verborgen neuronen.

3. Simulaties

Om na te gaan of het voorgestelde netwerk in staat is sequenties te categoriseren, zijn verschillende simulaties uitgevoerd. Bij iedere simulatie worden eerst toevoegingen op de algemene methode genoemd. Vervolgens worden de resultaten besproken. Tenslotte worden de resultaten van iedere simulatie in een eigen discussie besproken.

De eerste simulatie is het meest eenvoudig. Alle volgende simulaties zijn varianten van deze simulatie. De simulaties hebben tot doel eerdere simulaties te verbeteren of te verfijnen.

3.1. Simulatie 1: Categoriseren zonder gebruik van associaties tussen items

In deze simulatie wordt het netwerk op de meest eenvoudige wijze getest. Eerst worden de sequenties geleerd, waarna getest wordt of de sequenties ook kunnen worden gecategoriseerd. Bij het testen van categorisatie wordt geen rekening gehouden met de associaties die het netwerk tussen elementen legt bij het leren van de sequenties.

3.1.1. Methode

De algemene methode is aangehouden. Iedere verborgen laag neuronen bestond uit negen neuronen. Dit was aan de hand van informele analyse bepaald. Met vijf verborgen neuronen per laag was het netwerk niet in staat de sequenties te leren. Naarmate de verborgen lagen uit meer neuronen bestonden, was het netwerk hier beter en sneller toe staat in. Er is gekozen om het aantal verborgen neuronen te maximaliseren tot negen, omdat dit aansluit bij de vuistregel dat er niet meer neuronen in de verborgen lagen moeten zijn dan dat er inputneuronen zijn. Daarnaast leerde het netwerk met negen verborgen neuronen de sequenties het snelst wat voordelig is voor herhaalde uitvoering. Het netwerk werd getraind totdat een error van $1 \cdot 10^{-7}$ was bereikt.

Nadat het netwerk was getraind, werd een nieuwe inputset aangeboden om na te gaan of het netwerk ook in staat was de sequenties te categoriseren. In de algemene methode staat beschreven dat de input bestond uit een representatie van het tweede item van de sequenties en de bijbehorende draairichting. De representatie van het tweede item werd hier verkregen door voor iedere sequentie het tweede item via de inputneuronen aan te bieden. Er werd dus geen gebruik gemaakt van de verborgen inputneuronen.

3.1.2. Resultaten

Het aantal benodigde trainingscycli om bij het leren van de sequenties een error van $1 \cdot 10^{-7}$ te bereiken, was bij de eerste proefneming (*trial*) 20.474. Bij de overige twee proefnemingen lag het aantal cycli in dezelfde orde van grootte. Bij het testen of de verschillende sequenties inderdaad zijn geleerd, bleek de gegenereerde output 100% overeen te stemmen met de gewenste output. De opgeslagen activatiewaarden van de outputneuronen werden afgerond op twee decimalen.

De gegenereerde output is ook opgeslagen voor de inputset die bestaat uit het aanbieden van het tweede item van iedere sequentie en bijbehorende draairichting. Deze staat weergegeven in Tabel 2. Eerst werd bij alle drie de proefnemingen naar de situaties gekeken waar een differentiatie in richting vanwege de draairichting aanwezig is. Deze situaties zijn in Tabel 2 cursief gedrukt. Te zien is dat het netwerk bij slechts 1 van de 12 sequenties in staat was deze differentiatie te reproduceren. Dit uit zich in een activatiewaarde van outputneuron “noord” die desgewenst hoger of lager is dan de activatiewaarde van “zuid”. Bij 4 van de 24 sequenties bleek het netwerk in staat het gewenste derde element uit de sequentie te genereren.

3.1.3. Discussie

Uit de resultaten blijkt ten eerste dat het netwerk in staat is de sequenties te leren. Het netwerk lijkt echter niet in staat de sequenties te categoriseren. Dit blijkt uit het feit dat het

netwerk slechts bij 1 van de 12 relevante sequenties in staat blijkt de juiste eigenschap aan de categorieën linksom en rechtsom toe te schrijven (gaat in noordelijke dan wel zuidelijke richting na het tweede element). Ten slotte lijkt het netwerk niet goed in staat de sequenties te produceren als het figuurnummer niet als context wordt meegegeven. Dit blijkt uit het feit dat bij slechts 4 van de 24 sequenties het derde element van de sequentie correct werd gegenereerd na aanbieding van het tweede element als input.

Het is echter opvallend dat bijna iedere keer output “C” wordt gegenereerd als het tweede element van de sequentie en bijbehorende draairichting als input worden aangeboden. Als de geleerde sequenties uit Tabel 2 nogmaals worden bekeken, is een mogelijke verklaring te vinden. Een aantal inputelementen, komen namelijk meerdere keren voor bij een bepaalde draairichting; niet alleen als tweede element, maar ook als het element voor eindpunt “C”. Zo wordt in F1 het element “oost” bij een draairichting linksom gevolgd door “noordwest”. Bij F3 echter wordt het element “oost” bij een draairichting linksom gevolgd door het eindpunt “C”. Het is mogelijk dat bij deze ambiguïteit element “C” wordt verkozen boven “noordwest”, vanwege het feit dat element “C” het meest voorkomt in de sequenties en zo mogelijk een sterkere representatie heeft.

Om na te gaan wat de effecten zijn van ambiguïteit door het voorkomen van dezelfde items op verschillende locaties, wordt nog een simulatie gedaan. In simulatie 2 zal daarom het eindpunt “C” van alle sequenties niet worden mee geleerd, zodat deze bij ambiguïteit niet de sterkste representatie kan hebben.

Te zien is echter dat ook bij elementen waarbij deze ambiguïteit ontbreekt de gewenste output niet wordt gegenereerd bij het categoriseren van de sequenties. Een verklaring hiervoor is dat bij het testen of het netwerk in staat is te categoriseren geen rekening wordt gehouden met het feit dat het netwerk de verschillende elementen uit de sequentie aan elkaar associeert. Alleen het tweede element wordt namelijk aan het netwerk aangeboden om na te gaan of het

netwerk in staat is de sequenties te categoriseren. Om ook in de testfase gebruik te maken van de associatie tussen elementen moet het eerste element “C” worden aangeboden voordat het tweede element wordt aangeboden. Op deze manier krijgt het netwerk dezelfde input bij aanbieding van het tweede element als in de trainingsfase, namelijk het tweede element via de inputneuronen en een kopie van het eerste element via de verborgen inputneuronen. Ook bij de testfase zijn deze twee elementen dan aan elkaar geassocieerd.

Door gebruik te maken van de voorafgaande elementen bij het representeren van het tweede element van een sequentie, wordt ook de ambiguïteit tussen gelijke items op verschillende locaties vermeden. Dezelfde elementen worden namelijk door verschillende elementen vooraf gegaan en deze combinatie van elementen vormt nu de aanwijzing die het netwerk gebruikt om het volgende element op te roepen.

Om na te gaan wat het effect is van het voorkomen van ambiguïteit door gebruik te maken van het feit dat het netwerk elementen aan elkaar associeert, wordt ook nog een simulatie gedaan. In simulatie 3 worden nadat de sequenties zijn geleerd de eerste twee items van de sequentie achtereenvolgens aangeboden om na te gaan of het netwerk de sequenties ook heeft gecategoriseerd. Zo stemt ook de activatie van de verborgen inputneuronen overeen met de activatie van deze neuronen tijdens de trainingsfase.

Tabel 2

Output van het getrainde netwerk bij aanbieding van het tweede element van alle sequenties van simulatie 1

Input		Gegenereerde Output Trial 1					Gegenereerde Output Trial 2					Gegenereerde Output Trial 3					Gewenste Output
Draaiing	Element	Noord	Oost	Zuid	West	C	Noord	Oost	Zuid	West	C	Noord	Oost	Zuid	West	C	Element
L	*Oost*	*0,02*	*0,00*	*0,00*	*0,53*	*1,00*	*0,00*	*0,00*	*0,00*	*0,00*	*1,00*	*0,00*	*0,00*	*0,00*	*0,00*	*1,00*	*NoordWest*
R	NoordOost	0,00	0,00	0,00	0,05	0,98	0,00	0,99	1,00	0,05	0,00	0,00	0,00	0,00	0,00	1,00	ZuidOost
L	*West*	*0,00*	*0,00*	*0,00*	*0,00*	*1,00*	*0,00*	*0,00*	*0,00*	*0,05*	*1,00*	*0,00*	*0,00*	*0,00*	*0,00*	*1,00*	*ZuidOost*
R	ZuidWest	0,02	0,87	0,00	0,00	1,00	0,67	0,00	0,00	0,01	0,05	0,00	0,00	0,00	0,00	1,00	NoordWest
L	NoordWest	0,00	0,68	0,88	0,00	0,73	0,00	0,99	0,00	0,00	0,99	0,00	0,00	0,00	0,00	1,00	ZuidWest
R	*West*	*0,00*	*0,00*	*0,64*	*0,00*	*1,00*	*0,00*	*0,00*	*0,00*	*0,14*	*1,00*	*0,00*	*0,00*	*0,00*	*0,00*	*1,00*	*NoordOost*
L	ZuidOost	0,12	0,00	0,00	0,99	1,00	1,00	1,00	0,00	0,01	0,00	0,00	0,00	0,00	0,00	1,00	NoordOost
R	*Oost*	*0,00*	*0,00*	*0,00*	*0,00*	*1,00*	*0,00*	*0,00*	*0,00*	*0,00*	*1,00*	*0,00*	*0,00*	*0,00*	*0,00*	*1,00*	*ZuidWest*

Note. Door na te gaan of de activatie van “noord” of “zuid” desgewenst groter is én of de activatie van “oost” dan wel “west” desgewenst groter is, wordt bepaald of het gewenste derde is gegenereerd. Bij de cursief weergegeven rijen wordt nagegaan of het netwerk de sequenties juist heeft gecategoriseerd door na te gaan of de activatie van “noord” dan wel “zuid” desgewenst groter is.

3.2. Simulatie 2: Effecten van herhalingen zonder gebruik van associaties tussen items

In deze simulatie is geprobeerd de ambiguïteit te elimineren die wordt veroorzaakt doordat bepaalde elementen herhaald voorkomen, maar door verschillende elementen worden gevolgd. Dit is gedaan door de verschillende sequenties zonder eindpunt te leren, aangezien het eindpunt deze ambiguïteit veroorzaakte. Bij het testen of het netwerk in staat is de sequenties te categoriseren is nog steeds geen rekening gehouden met de associaties tussen elementen die het netwerk legt tijdens de trainingsfase.

3.2.1. Methode

In de tweede simulatie werden de sequenties zonder eindpunt geleerd. Iedere verborgen laag bestond uit negen neuronen. Verder waren er vier outputneuronen, omdat het eindpunt “C” niet meer nodig was. Het netwerk werd getraind, totdat een error van $1 \cdot 10^{-7}$ was bereikt. Om na te gaan of het netwerk in staat is de sequenties te categoriseren, dient het tweede element van iedere sequentie te worden gerepresenteerd. Dit is, evenals bij simulatie 1, gedaan door het tweede element als input aan het netwerk aan te bieden. Verder is de algemene methode aangehouden.

3.2.2. Resultaten

Het aantal benodigde trainingscycli om bij het leren van de sequenties een error van $1 \cdot 10^{-7}$ te bereiken, was bij de eerste proefneming 17.994. Bij de overige twee proefnemingen lag het aantal cycli in dezelfde orde van grootte. Bij het testen of de verschillende sequenties inderdaad waren geleerd, bleek de gegenereerde output 100% overeen te stemmen met de gewenste output.

De gegenereerde output is ook opgeslagen voor de inputset die bestaat uit het aanbieden van het tweede item van iedere sequentie en bijbehorende draairichting. Deze staat weergegeven in Tabel 3. Eerst wordt bij alle drie de proefnemingen naar de situaties gekeken

waar een differentiatie in richting vanwege de draairichting aanwezig is. Deze situaties zijn in Tabel 3 cursief weergegeven. Te zien is dat het netwerk bij 8 van de 12 sequenties in staat was deze differentiatie te reproduceren. In de eerste trial was het netwerk hier bij alle vier de sequenties toe in staat. Bij 15 van de 24 sequenties bleek het netwerk in staat het gewenste derde element uit de sequentie te genereren.

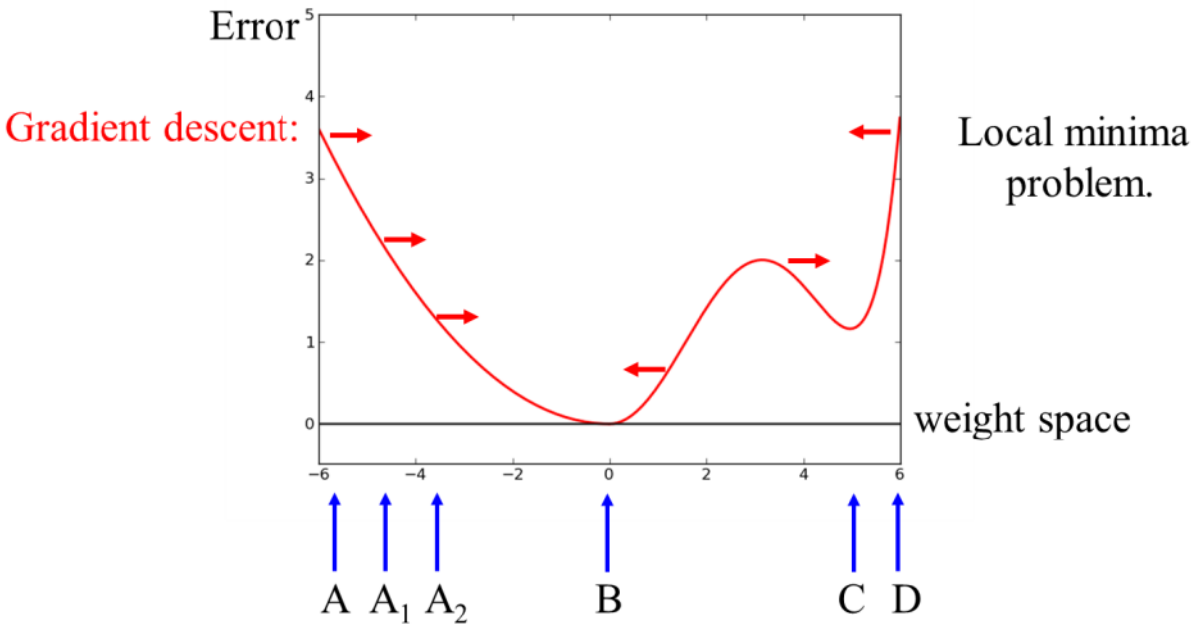
3.2.3. Discussie

Deze resultaten zijn beduidend beter dan die van de eerste simulatie. Ten eerste lijkt het netwerk de sequenties sneller te leren, wat waarschijnlijk komt doordat alle sequenties een element korter zijn geworden. Daarnaast lijkt het netwerk beter in staat de sequenties te produceren als het figuurnummer niet als context wordt meegegeven. Dit blijkt uit het feit dat in deze simulatie bij 15 van de 24 sequenties het derde element van de sequentie correct werd gegenereerd na aanbieding van het tweede element als input in tegenstelling tot 4 van de 24 sequenties bij de vorige simulatie.

Het netwerk lijkt ten slotte beter in staat de sequenties te categoriseren. Bij deze simulatie was het netwerk namelijk bij 8 van de 12 sequenties in staat de juiste eigenschap horende bij de categorieën linksom of rechtsom te produceren. Bij de eerste trial maakte het netwerk zelfs helemaal geen fouten. Dit duidt erop dat het netwerk soms wel in staat is de sequenties te categoriseren naar draairichting en soms niet.

Het feit dat de verschillende trials verschillende uitkomsten opleveren, komt doordat de leermethode *backpropagation* gevoelig is voor de initiële staat van het netwerk (Kolen & Pollack, 1990). In de initiële staat van alle proefnemingen hebben de connectiegewichten willekeurige waardes. Zoals eerder staat beschreven, wordt er bij *backpropagation* gekeken waar de grootste error vandaan komt en wordt deze error verkleind door de connectiegewichten aan te passen. Dit proces van het aanpassen van de connectiegewichten, zodat de error steeds kleiner wordt, wordt *gradient descent* genoemd (Van der Velde, 2011b).

Het aanpassen van de connectiegewichten herhaalt zich, totdat de error niet meer kleiner wordt, oftewel totdat er een minimum in de errorfunctie is bereikt. Dit kan echter een locaal minimum zijn in plaats van het globale minimum en hoeft daarom niet de beste oplossing te zijn. In Figuur 8 is te zien dat *gradient descent* tot gevolg kan hebben dat een locaal minimum van de errorfunctie wordt bereikt in plaats van het globale minimum. Welk minimum wordt bereikt hangt af van de set connectiegewichten waarmee wordt gestart. Sommige sets connectiegewichten liggen namelijk al dicht bij een goede oplossing, terwijl andere daar ver vandaan liggen (Lee, 2010). Een leermethode die hier op in speelt is *deep learning* en wordt in de algemene discussie verder besproken (Hinton & Salakhutdinov, 2006).



Figuur 8. De invloed van *gradient descent* op het vinden van een minimum. Als vanuit punt "A" de error alsmaar wordt verkleind, wordt het globale minimum "B" van dit bereik van de errorfunctie bereikt. Start men vanaf "D", dan wordt het locale minimum "C" bereikt: Uit: "Learning in Membrain", door Van der Velde, F., 2011b, p.5.

In deze simulatie is ambiguïteit, veroorzaakt door het vaker optreden van bepaalde item-draairichting combinaties, voorkomen door de item-draairichting combinaties die in strijd zijn met de gewenste output te verwijderen. In een optimale situatie is het netwerk echter ook in staat met zulke ambiguïteiten om te gaan. Daarom wordt in de volgende simulatie gekeken of het netwerk (ook) in staat is de sequenties te categoriseren als alleen een

andere representatie van het tweede element van de sequenties aan het getrainde netwerk wordt aangeboden. Deze representatie bestaat uit het achtereenvolgens aanbieden van het eerste en het tweede element van de sequentie, zodat ook gebruik wordt gemaakt van de capaciteit van het netwerk om elementen in de sequenties aan elkaar te associëren.

Tabel 3

Output van het getrainde netwerk bij aanbieding van het tweede element van alle sequenties van simulatie 2

Input		Gegenereerde Output Trial 1				Gegenereerde Output Trial 2				Gegenereerde Output Trial 3				Gewenste Output
Draaiing	Element	Noord	Oost	Zuid	West	Noord	Oost	Zuid	West	Noord	Oost	Zuid	West	Element
L	*Oost*	*0,66*	*1,00*	*0,00*	*0,00*	*0,00*	*0,00*	*0,01*	*1,00*	*0,96*	*0,05*	*0,00*	*0,95*	*NoordWest*
R	NoordOost	0,00	1,00	1,00	0,00	0,00	0,89	0,00	0,11	0,00	0,00	1,00	1,00	ZuidOost
L	*West*	*0,01*	*0,99*	*0,99*	*0,01*	*0,00*	*0,99*	*0,00*	*0,01*	*0,00*	*1,00*	*0,21*	*0,00*	*ZuidOost*
R	ZuidWest	0,99	0,00	0,00	1,00	0,02	0,04	0,00	0,96	1,00	0,00	0,00	1,00	NoordWest
L	NoordWest	0,00	0,00	1,00	1,00	0,00	0,01	0,00	0,99	0,00	0,99	1,00	0,01	ZuidWest
R	*West*	*1,00*	*1,00*	*0,00*	*0,00*	*0,59*	*1,00*	*0,00*	*0,00*	*0,70*	*1,00*	*0,02*	*0,00*	*NoordOost*
L	ZuidOost	1,00	1,00	0,00	0,00	1,00	1,00	0,00	0,00	1,00	0,99	0,00	0,01	NoordOost
R	*Oost*	*0,00*	*0,01*	*0,99*	*0,99*	*0,01*	*0,00*	*0,00*	*1,00*	*0,11*	*0,00*	*0,00*	*1,00*	*ZuidWest*

Note. Door na te gaan of de activatie van “noord” of “zuid” desgewenst groter is én of de activatie van “oost” dan wel “west” desgewenst groter is, wordt bepaald of het gewenste derde is gegenereerd. Bij de cursief weergegeven rijen wordt nagegaan of het netwerk de sequenties juist heeft gecategoriseerd door na te gaan of de activatie van “noord” dan wel “zuid” desgewenst groter is.

3.3. Simulatie 3: Categoriseren met gebruik van associaties tussen items

In deze simulatie is eveneens geprobeerd de ambiguïteit te elimineren die wordt veroorzaakt doordat bepaalde elementen herhaald voorkomen, maar door verschillende elementen worden gevolgd. In deze simulatie is dat echter gedaan door ook tijdens het testen van categorisatie gebruik te maken van de associaties tussen elementen die het netwerk legt tijdens de trainingsfase. Dit wordt gedaan door bij de testfase ook het eerste en het tweede element achtereenvolgens aan te bieden. Hierdoor krijgt het netwerk bij input van het tweede element via de verborgen inputneuronen ook een kopie van het eerste element aangeboden net als in de trainingsfase.

3.3.1. Methode

De algemene methode was aangehouden. Iedere verborgen laag neuronen bestond uit negen neuronen en het netwerk werd getraind totdat een error van $1 \cdot 10^{-7}$ was bereikt. Nadat het netwerk was getraind, werd weer een nieuwe inputset aangeboden om na te gaan of het netwerk in staat is de sequenties te categoriseren.

In de algemene methode staat beschreven dat de input bestaat uit een representatie van het tweede item van de sequenties en de bijbehorende draairichting. De representatie van het tweede item werd hier verkregen door voor iedere sequentie achtereenvolgens het eerste en het tweede item via de inputneuronen aan te bieden. Op deze wijze werd bij de aanbieding van het tweede item via de inputneuronen gelijktijdig de representatie van het eerste item via de verborgen inputneuronen aan het netwerk aangeboden. Dit is in tegenstelling tot eerdere representaties van het tweede element, waar geen rekening werd gehouden met de associaties met eerdere items en dus geen gebruik werd gemaakt van de verborgen inputneuronen.

3.3.2. Resultaten

Het aantal benodigde trainingscycli om bij het leren van de sequenties een error van $1 \cdot 10^{-7}$ te bereiken, was bij de eerste proefneming 21.084. Bij de overige twee proefnemingen

lag het aantal cycli in dezelfde orde van grootte. Bij het testen of de verschillende sequenties inderdaad zijn geleerd, bleek de gegenereerde output 100% overeen te stemmen met de gewenste output.

De gegenereerde output is ook opgeslagen voor de inputset die bestaat uit het aanbieden van achtereenvolgens het eerste en het tweede item van iedere sequentie en bijbehorende draairichting. Deze staat weergegeven in Tabel 4. De gegenereerde output bij input “C” is niet weergegeven, omdat deze niet relevant is. Eerst wordt bij alle drie de proefnemingen naar de situaties gekeken waar een differentiatie in richting vanwege de draairichting aanwezig is. Deze situaties zijn in Tabel 4 cursief weergegeven. Te zien is dat het netwerk bij 8 van de 12 sequenties in staat was deze differentiatie te reproduceren. In de eerste trial was het netwerk hier bij alle vier de sequenties toe in staat. Bij 14 van de 24 sequenties bleek het netwerk in staat het gewenste derde element uit de sequentie te genereren.

3.3.3. Discussie

De resultaten van deze simulatie komen overeen met de resultaten van simulatie 2. Dit toont aan dat het netwerk met de ambiguïteit veroorzaakt door herhaalde item-draairichting combinaties ook kan omgaan door gebruik te maken van de recurrente connecties van het netwerk. Deze zorgen er namelijk voor dat de representatie van een element uit een sequentie ook afhangt van de voorgaande elementen. De representatie van bijvoorbeeld element “oost” op de tweede positie in een sequentie is dan anders dan de representatie van het element “oost” op de vierde positie in een sequentie.

Net als bij simulatie 2 heeft het netwerk in deze simulatie bij de eerste proefneming ook geen fouten gemaakt in het produceren van de juiste eigenschap horende bij de categorieën linksom of rechtsom. Dit duidt er wederom op dat het netwerk bij bepaalde

randomisaties van de connectiegewichten wel in staat lijkt te zijn de sequenties te categoriseren.

De werkwijze uit deze simulatie en simulatie 2 zullen in de volgende simulatie worden gecombineerd om na te gaan of deze werkwijzen elkaar wellicht completeren. In deze simulatie zullen de sequenties dus zonder eindpunt worden geleerd en bij de representatie van het tweede element van de sequentie wordt rekening gehouden met de associatie met eerdere elementen.

Tabel 4

Output van het getrainde netwerk bij aanbieding van het tweede element van alle sequenties van simulatie 3

Input		Gegenereerde Output Trial 1					Gegenereerde Output Trial 2					Gegenereerde Output Trial 3					Gewenste Output
Draaiing	Element	Noord	Oost	Zuid	West	C	Noord	Oost	Zuid	West	C	Noord	Oost	Zuid	West	C	Element
L	*Oost*	*1,00*	*0,00*	*0,00*	*1,00*	*0,00*	*0,96*	*0,00*	*0,00*	*1,00*	*0,00*	*0,69*	*0,00*	*0,91*	*0,98*	*0,00*	*NoordWest*
R	NoordOost	0,00	1,00	1,00	0,00	0,00	0,00	1,00	1,00	0,00	0,00	0,00	0,92	1,00	0,01	0,00	ZuidOost
L	*West*	*0,92*	*0,69*	*1,00*	*1,00*	*0,00*	*0,00*	*0,82*	*0,00*	*0,00*	*0,99*	*0,00*	*1,00*	*0,00*	*0,00*	*0,00*	*ZuidOost*
R	ZuidWest	1,00	0,00	0,00	0,83	0,02	1,00	1,00	0,72	1,00	0,00	1,00	0,00	0,00	1,00	0,00	NoordWest
L	NoordWest	0,00	0,45	1,00	0,98	0,00	0,00	0,00	1,00	1,00	0,00	0,00	0,00	1,00	1,00	0,00	ZuidWest
R	*West*	*1,00*	*1,00*	*0,00*	*0,92*	*0,00*	*0,00*	*1,00*	*1,00*	*0,00*	*0,00*	*0,15*	*0,00*	*0,00*	*0,55*	*0,07*	*NoordOost*
L	ZuidOost	1,00	0,94	0,00	1,00	0,00	1,00	0,99	0,00	0,00	0,00	0,99	1,00	0,00	0,00	0,00	NoordOost
R	*Oost*	*0,00*	*1,00*	*0,56*	*0,00*	*0,00*	*0,00*	*0,98*	*0,99*	*0,00*	*0,00*	*0,00*	*0,00*	*0,23*	*1,00*	*0,00*	*ZuidWest*

Note. Door na te gaan of de activatie van “noord” of “zuid” desgewenst groter is én of de activatie van “oost” dan wel “west” desgewenst groter is, wordt bepaald of het gewenste derde is gegenereerd. Bij de cursief weergegeven rijen wordt nagegaan of het netwerk de sequenties juist heeft gecategoriseerd door na te gaan of de activatie van “noord” dan wel “zuid” desgewenst groter is.

3.4. Simulatie 4: Effecten van herhalingen met gebruik van associaties tussen items

In deze simulatie wordt nagegaan of de methodes die zijn gebruikt in simulatie 2 en in simulatie 3 om met herhaalde elementen in sequenties om te gaan elkaar completeren of dat beide methodes overlappen. De methodes van simulatie 2 en 3 worden daarom samengevoegd.

3.4.1. Methode

De algemene methode is grotendeels aangehouden. De sequenties zijn echter zonder eindpunt geleerd. Iedere verborgen laag neuronen bestond uit negen neuronen en er waren maar vier output neuronen, omdat ouputneuron “C” in deze simulatie overbodig was. Het netwerk werd getraind totdat een error van $1 \cdot 10^{-7}$ was bereikt. Nadat het netwerk was getraind, werd weer een nieuwe inputset aangeboden om na te gaan of het netwerk in staat is de sequenties te categoriseren.

In de algemene methode staat beschreven dat de input bestaat uit een representatie van het tweede item van de sequenties en de bijbehorende draairichting. De representatie van het tweede item werd hier verkregen door voor iedere sequentie achtereenvolgens het eerste en het tweede item via de inputneuronen aan te bieden. Op deze wijze werd bij de aanbieding van het tweede item via de inputneuronen gelijktijdig de representatie van het eerste item via de verborgen inputneuronen aan het netwerk aangeboden.

3.4.2. Resultaten

Het aantal benodigde trainingscycli om bij het leren van de sequenties een error van $1 \cdot 10^{-7}$ te bereiken, was bij de eerste proefneming 16.873. Bij de overige twee proefnemingen lag het aantal cycli in dezelfde orde van grootte. Bij het testen of de verschillende sequenties inderdaad zijn geleerd, bleek de gegenereerde output 100% overeen te stemmen met de gewenste output.

De gegenereerde output is ook opgeslagen voor de inputset die bestaat uit het achtereenvolgens aanbieden van het eerste en tweede item van iedere sequentie en bijbehorende draairichting. Deze staat weergegeven in Tabel 5. De gegenereerde output bij input “C” is niet weergegeven, omdat deze niet relevant is. Eerst wordt bij alle drie de trials naar de situaties gekeken waar een differentiatie in richting vanwege de draairichting aanwezig is. Deze situaties zijn in Tabel 5 cursief weergegeven. Te zien is dat het netwerk bij 10 van de 12 sequenties in staat was deze differentiatie te reproduceren. In de eerste trial was het netwerk hier bij alle vier de sequenties toe in staat. Bij 16 van de 24 sequenties bleek het netwerk in staat het gewenste derde element uit de sequentie te genereren.

3.4.3. Discussie

Deze resultaten komen redelijk overeen met de resultaten van de tweede en derde simulatie. Dit duidt erop dat de simulaties elkaar niet completeren. Net als bij simulatie 2 en simulatie 3 werden bij één proefneming de juiste eigenschappen van de categorieën linksom en rechtsom geproduceerd bij de vier relevante sequenties. Zoals eerder beschreven toont dit aan dat het netwerk bij bepaalde randomisaties in staat is de sequenties te categoriseren.

Daarnaast is het netwerk ongeveer even vaak in staat het correcte derde element van de sequentie te produceren na aanbieding van een representatie van het tweede element als bij simulatie 2 en simulatie 3 het geval was. Dit duidt erop dat de netwerken even goed in staat zijn de sequenties te reproduceren als een gedeelte van het netwerk niet meer functioneert, namelijk als het figuurnummer niet meer wordt aangeboden.

Een optimale functionaliteit is echter nog niet bereikt. Wellicht komt dit door de wijze waarop de sequenties zijn geleerd. Het gebruikte programma, MemBrain, is namelijk niet in staat de verschillende sequenties in willekeurige volgorde te leren. Alle sequenties worden als het ware als één lange sequentie geleerd. De enige manier waarop te herkennen is dat een nieuwe sequentie is begonnen, is de veranderde context. Dit heeft tot mogelijk gevolg dat het

tweede item van een sequentie niet alleen geassocieerd wordt met het eerste item van die sequentie, maar ook met een “x” aantal items van de vorige sequentie. Het interne geheugen van het netwerk zou daarom na iedere sequentie gewist moeten worden.

Een manier om het interne geheugen van het netwerk te wissen is door meerdere keren achter elkaar hetzelfde element aan te bieden. Als het element vaak genoeg wordt aangeboden, zal de invloed van het vorige element op de activatiewaarden van de eerste verborgen laag neuronen geen rol meer spelen. De eerste stappen van dit proces gaan als volgt. Eerst wordt element “A” aan het netwerk aangeboden. Dit element wordt in de eerste laag verborgen neuronen gerepresenteerd en er wordt een kopie van deze representatie gemaakt in de verborgen inputneuronen. Vervolgens wordt element “B” aan het netwerk aangeboden. De eerste laag verborgen inputneuronen representeren nu een combinatie van “B” en de kopie van “A”. Deze combinatie, “AB”, wordt nu gekopieerd naar de verborgen inputneuronen. Als nogmaals “B” wordt aangeboden, representeert de eerste laag neuronen een combinatie van “B” en de kopie van “AB”, namelijk “ABB”. Zo is te zien dat de representatie van “A” een steeds minder belangrijke rol krijgt. Op deze manier worden de elementen vóór het herhaald aanbieden van een element “X” als het ware uit het geheugen gestoten. In de volgende simulatie zal deze methode worden toegepast.

Tabel 5

Output van het getrainde netwerk bij aanbieding van het tweede element van alle sequenties van simulatie 4

Input		Gegenereerde Output Trial 1				Gegenereerde Output Trial 2				Gegenereerde Output Trial 3				Gewenste Output
Draaiing	Element	Noord	Oost	Zuid	West	Noord	Oost	Zuid	West	Noord	Oost	Zuid	West	Element
L	*Oost*	*0,99*	*0,96*	*0,76*	*0,04*	*1,00*	*0,00*	*0,00*	*1,00*	*0,64*	*0,00*	*0,00*	*1,00*	*NoordWest*
R	NoordOost	0,00	1,00	1,00	0,00	0,00	1,00	1,00	0,00	0,00	1,00	1,00	0,00	ZuidOost
L	*West*	*0,00*	*1,00*	*0,99*	*0,00*	*0,00*	*1,00*	*1,00*	*0,00*	*0,00*	*1,00*	*0,00*	*0,00*	*ZuidOost*
R	ZuidWest	1,00	0,01	0,00	0,99	1,00	1,00	0,00	0,00	1,00	0,00	0,00	1,00	NoordWest
L	NoordWest	0,00	0,00	1,00	1,00	0,00	0,87	1,00	0,13	0,00	0,90	1,00	0,10	ZuidWest
R	*West*	*1,00*	*1,00*	*0,00*	*0,00*	*0,00*	*1,00*	*1,00*	*0,00*	*0,05*	*1,00*	*0,00*	*0,00*	*NoordOost*
L	ZuidOost	1,00	1,00	0,00	0,00	1,00	0,78	0,00	0,22	1,00	0,99	0,00	0,01	NoordOost
R	*Oost*	*0,00*	*0,83*	*0,92*	*0,14*	*0,00*	*0,00*	*0,37*	*1,00*	*0,00*	*0,96*	*1,00*	*0,04*	*ZuidWest*

Note. Door na te gaan of de activatie van “noord” of “zuid” desgewenst groter is én of de activatie van “oost” dan wel “west” desgewenst groter is, wordt bepaald of het gewenste derde is gegenereerd. Bij de cursief weergegeven rijen wordt nagegaan of het netwerk de sequenties juist heeft gecategoriseerd door na te gaan of de activatie van “noord” dan wel “zuid” desgewenst groter is.

3.5. Simulatie 5: Effecten van het wissen van het interne geheugen (C)

In deze simulatie is geprobeerd het interne geheugen van het netwerk na iedere sequentie te wissen, zodat het leren van de ene sequentie geen invloed heeft op het leren van de volgende sequentie. Dit is nodig, omdat het in MemBrain niet mogelijk is losse sequenties in willekeurige volgorde te leren. Het eerste element van de tweede sequentie wordt daarom via de recurrente verbindingen geassocieerd aan de laatste "x" elementen van de vorige sequentie. Door het interne geheugen na iedere sequentie te wissen, kunnen de sequenties toch apart van elkaar geleerd worden.

3.5.1. Methode

Om het interne geheugen te wissen, is aan het einde van iedere sequentie nog vier keer het element "C" toegevoegd. Dit zou ertoe moeten leiden dat bij aanbieding van het eerste element van de volgende sequentie, de verborgen inputneuronen slechts het element "C" representeren en de vorige elementen geen rol meer spelen. Het element "C" dat de start van de volgende sequentie representeert, is te herkennen aan de andere context die gelijktijdig actief is.

Verder is de algemene methode aangehouden. Iedere verborgen laag neuronen bestond uit negen neuronen en het netwerk werd getraind totdat een error van $1 \cdot 10^{-7}$ was bereikt. Nadat het netwerk was getraind, werd weer een nieuwe inputset aangeboden om na te gaan of het netwerk in staat is de sequenties te categoriseren. Deze inputset bestond uit de representatie van het tweede item door achtereenvolgens het eerste en het tweede item van iedere sequentie aan te bieden samen met de draairichting.

3.5.2. Resultaten

Het netwerk bleek niet in staat de nieuwe sequenties compleet te leren als de recurrente connecties niet werden getraind. Bij verschillende pogingen om het netwerk te

trainen, werd al snel een stabiele error van ongeveer 0.06 bereikt. Ook als het trainingsproces werd voortgezet, werd de error niet lager. Als werd gekeken hoe het netwerk presteerde bij deze error, viel op dat het netwerk de meeste opvolgingen in de sequenties goed leerde. Het probleem ligt echter bij de start van iedere sequentie. Als bij een input van startpunt "C" het tweede element van de sequentie als output werd verwacht, werd bij iedere sequentie de verkeerde output gegenereerd. De gegenereerde output was in alle gevallen "C". Zie Bijlage A voor de tabel met input- en outputwaarden,.

Ook als de recurrente connecties werden geleerd, had het netwerk moeite de sequenties te leren. Slechts bij sommige randomisaties van de connectiegewichten was het netwerk hier toe in staat. Om toch de invloed van het wissen van het interne geheugen op het leren en categoriseren van de sequenties te bekijken, zijn drie trials bekeken waar het netwerk wel in staat was de sequenties te leren. Hierbij werden dus ook de recurrente verbindingen getraind.

Het aantal benodigde trainingscycli om bij het leren van de sequenties een error van $1 \cdot 10^{-5}$ te bereiken, was bij de eerste proefneming 17.803. Bij de overige twee proefnemingen lag het aantal cycli in dezelfde orde van grootte. Bij het testen of de verschillende sequenties inderdaad zijn geleerd, bleek de gegenereerde output 100% overeen te stemmen met de gewenste output.

De gegenereerde output is ook opgeslagen voor de inputset die bestaat uit het achtereenvolgens aanbieden van het eerste en tweede item van iedere sequentie met bijbehorende draairichting. Deze staat weergegeven in Tabel 6. De gegenereerde output bij input "C" is niet weergegeven, omdat deze niet relevant is. Eerst wordt bij alle drie de proefnemingen naar de situaties gekeken waar een differentiatie in richting vanwege de draairichting aanwezig is. Deze situaties zijn in Tabel 6 cursief weergegeven. Te zien is dat het netwerk bij 4 van de 12 sequenties in staat was deze differentiatie te reproduceren. Bij 10

van de 24 sequenties bleek het netwerk in staat het gewenste derde element uit de sequentie te genereren.

3.5.3. Discussie

Het is opvallend dat het netwerk moeite heeft met het leren van deze sequenties en hier zonder het trainen van de recurrente verbindingen niet toe in staat lijkt. Als naar Bijlage A wordt gekeken, is te zien waar het probleem ligt. Iedere keer als het element dat startpunt "C" volgt als output wordt verwacht, wordt weer "C" gegenereerd.

Dat het netwerk problemen heeft met het element "C" is ook terug te zien als de recurrente verbindingen extra worden geleerd. Dit netwerk presteert namelijk relatief slecht als het na aanbieding van het tweede element van de sequenties het derde element moet produceren. Te zien is namelijk dat weer vaak de output "C" wordt gegenereerd in plaats van het gewenste derde element. Met name bij proefneming 3 is dit goed te zien.

Een verklaring is dat het netwerk moeite heeft het onderscheid te leren tussen "C" als startpunt van de sequentie en "C" als methode om het interne geheugen te wissen. Het netwerk moet hier namelijk kunnen onderscheiden dat wanneer input "C" wordt gebruikt om het interne geheugen te wissen, output "C" wordt verwacht. Als input "C" het startpunt van de sequentie betekent, wordt deze echter gevolgd door een bepaalde windrichting. Dit zorgt voor ambiguïteit. Deze ambiguïteit verschilt echter met de ambiguïteit bij simulatie 1. Bij simulatie 1 werd namelijk geen gebruik gemaakt van het feit dat het netwerk in staat is verschillende situaties te onderscheiden doordat een bepaalde input tegelijk wordt aangeboden met een representatie van de voorgaande items.

Bij deze simulatie wordt het item "C" echter vaak herhaald wat de problemen oplevert. In het geval dat men bij de laatste "C" is aangekomen die het interne geheugen dient te wissen, wordt input "C" tegelijk aangeboden met de representatie van "C", omdat aan dit item meerdere malen "C" is voorafgegaan. Bij deze input wordt output "C" verwacht, het startpunt

van de volgende sequentie. In het geval dat we bij dit startpunt zijn aangekomen, wordt input “C” weer tegelijk aangeboden met de representatie van “C”, omdat ook aan dit startpunt meerdere malen “C” is voorafgegaan. Nu wordt echter een bepaalde windrichting als output verwacht. Dus in dit geval helpen de voorafgaande elementen niet bij het onderscheiden van de twee situaties.

Het feit dat het netwerk niet kan omgaan met veel herhalingen achter elkaar is een ernstige tekortkoming. Voor het nagaan of het netwerk in staat is sequenties te categoriseren, kan dit probleem echter ontlopen worden door een ander element dan “C” te gebruiken om het interne geheugen te wissen. Als bijvoorbeeld een nieuw element “X” wordt gebruikt, is het niet erg als het netwerk in een *loop* blijft “hangen”, omdat bij het wissen van het interne geheugen input “X” alleen maar wordt gevolgd door output “X”. Als dit een aantal keer is gedaan, kan gewoon gestart worden met de nieuwe sequentie en input “C”.

Tabel 6

Output van het getrainde netwerk bij aanbieding van het tweede element van alle sequenties van simulatie 5

Input		Gegenereerde Output Trial 1					Gegenereerde Output Trial 2					Gegenereerde Output Trial 3					Gewenste Output
Draaiing	Element	Noord	Oost	Zuid	West	C	Noord	Oost	Zuid	West	C	Noord	Oost	Zuid	West	C	Element
L	*Oost*	*1,00*	*0,05*	*0,13*	*1,00*	*0,00*	*0,04*	*0,32*	*0,68*	*0,56*	*0,00*	*1,00*	*0,15*	*0,00*	*1,00*	*0,00*	*NoordWest*
R	NoordOost	0,00	0,86	1,00	0,23	0,00	0,00	1,00	1,00	0,00	0,00	0,00	1,00	1,00	0,00	0,00	ZuidOost
L	*West*	*0,00*	*0,67*	*1,00*	*0,17*	*0,00*	*0,00*	*0,00*	*0,00*	*0,00*	*1,00*	*0,00*	*0,00*	*0,00*	*0,00*	*1,00*	*ZuidOost*
R	ZuidWest	1,00	0,97	0,00	0,32	0,00	0,93	0,20	0,00	0,27	0,00	0,00	0,00	0,00	0,00	1,00	NoordWest
L	NoordWest	0,00	0,95	1,00	0,12	0,00	0,00	0,00	0,85	0,38	0,00	0,00	0,00	0,00	0,00	1,00	ZuidWest
R	*West*	*0,99*	*0,99*	*0,14*	*0,06*	*0,00*	*0,00*	*0,00*	*0,00*	*0,00*	*0,99*	*0,00*	*0,00*	*0,00*	*0,00*	*1,00*	*NoordOost*
L	ZuidOost	1,00	1,00	0,00	0,00	0,00	0,87	0,00	0,00	0,99	0,00	0,00	0,00	0,00	0,00	1,00	NoordOost
R	*Oost*	*0,00*	*0,00*	*0,00*	*0,00*	*1,00*	*0,00*	*0,00*	*0,00*	*0,01*	*0,99*	*0,00*	*0,00*	*0,00*	*0,01*	*1,00*	*ZuidWest*

Note. Door na te gaan of de activatie van “noord” of “zuid” desgewenst groter is én of de activatie van “oost” dan wel “west” desgewenst groter is, wordt bepaald of het gewenste derde is gegenereerd. Bij de cursief weergegeven rijen wordt nagegaan of het netwerk de sequenties juist heeft gecategoriseerd door na te gaan of de activatie van “noord” dan wel “zuid” desgewenst groter is.

3.6. Simulatie 6: Effecten van het wissen van het interne geheugen (X)

In deze simulatie is op een andere wijze geprobeerd het interne geheugen van het netwerk na iedere sequentie te wissen in verband met problemen die de methode van simulatie 5 met zich meebracht. Het netwerk blijkt namelijk niet in staat om te gaan met het aanbieden van een bepaald element meerdere keren achter elkaar. Het netwerk komt dan in een *loop* terecht en kan geen andere output meer genereren. Zodra het interne geheugen echter wordt gewist met een element dat niet in de sequenties voorkomt, hoeft het netwerk niet uit deze *loop* te komen om toch goed te functioneren. Nadat het interne geheugen is gewist, krijgt het netwerk namelijk weer het eerste element van de volgende sequentie als input en moet op basis van deze input het tweede element oproepen.

3.6.1. Methode

Aan het einde van iedere sequentie is in deze simulatie nog vier keer het nieuwe element “X” toegevoegd. Omdat het element “X” alleen wordt gebruikt om het interne geheugen te wissen en niet in de routes zelf voorkomt, is het voor het netwerk wellicht makkelijker om te herkennen dat een nieuwe sequentie wordt gestart. De start van de volgende sequentie, is nu niet alleen te herkennen aan de andere context die gelijktijdig actief is, maar ook het startelement, “C”, verschilt van het elementen dat hier direct aan voorafgaat.

Verder is de algemene methode aangehouden. Iedere verborgen laag neuronen bestond uit negen neuronen en het netwerk werd getraind totdat een error van $1 \cdot 10^{-7}$ was bereikt. Aan de input- en outputneuronen werd een extra neuron toegevoegd dat het element “X” representeerde. Dit leidde tot een totaal aantal van zes neuronen in deze lagen. Nadat het netwerk was getraind, werd weer een nieuwe inputset aangeboden om na te gaan of het netwerk in staat is de sequenties te categoriseren. Deze inputset bestond uit de representatie van het tweede item door achtereenvolgens het eerste en het tweede item van iedere sequentie aan te bieden samen met de draairichting.

3.6.2. Resultaten

Het aantal benodigde trainingscycli om bij het leren van de sequenties een error van $1 \cdot 10^{-7}$ te bereiken, was bij de eerste proefneming 63.677. Bij de overige twee proefnemingen lag het aantal cycli in dezelfde orde van grootte. Bij het testen of de verschillende sequenties inderdaad zijn geleerd, bleek de gegenereerde output 100% overeen te stemmen met de gewenste output.

De gegenereerde output is ook opgeslagen voor de inputset die bestaat uit het achtereenvolgens aanbieden van het eerste en tweede item van iedere sequentie met bijbehorende draairichting. Deze staat weergegeven in Tabel 7. De gegenereerde output bij input “C” is niet weergegeven, omdat deze niet relevant is. Eerst wordt bij alle drie de trials naar de situaties gekeken waar een differentiatie in richting vanwege de draairichting aanwezig is. Deze situaties zijn in Tabel 7 cursief weergegeven. Te zien is dat het netwerk bij 8 van de 12 sequenties in staat was deze differentiatie te reproduceren. In de derde proefneming was het netwerk hier bij alle vier de sequenties toe in staat. Bij 20 van de 24 sequenties bleek het netwerk in staat het gewenste derde element uit de sequentie te genereren. Dit is dus voor alle sequenties waar het categoriseren van de sequenties geen rol speelt het geval .

3.6.3. Discussie

De resultaten van deze simulatie zien er relatief goed uit. Ten eerste is te zien dat het netwerk de sequenties kan leren zonder ook de recurrente verbindingen te moeten leren. Daarnaast valt op dat niet meer “C” of “X” als output wordt gegenereerd na aanbieding van het tweede element van de sequenties. Dit toont aan dat het netwerk inderdaad geen problemen meer heeft met het grote aantal herhalingen van één item achter elkaar. Het netwerk blijft bij het herhalen van input “X” en output “X” waarschijnlijk nog wel ineen *loop* hangen, maar voor de werking van het netwerk is het niet belangrijk dat het hieruit komt.

Daarnaast lijkt het netwerk, net als eerdere simulaties, bij bepaalde randomisaties van de connectiegewichten in staat de sequenties te categoriseren. Dit uit zich in het feit dat bij één proefneming het netwerk in staat was bij alle relevante sequenties de juiste eigenschap bij de categorieën linksom of rechtsom te produceren.

Tenslotte is dit netwerk goed in staat de geleerde sequenties te reproduceren als een gedeelte van het netwerk niet meer functioneert. Op dit gebied heeft deze simulatie de beste resultaten. Bij alle sequenties die niet relevant waren voor het categoriseren van sequenties, werd namelijk correct het derde element gegenereerd bij aanbieding van achtereenvolgens het eerste en het tweede element, samen met de draairichting. Dit kan betekenen dat het inderdaad belangrijk is het interne geheugen van het netwerk te wissen na iedere sequentie, zodat de representatie van het tweede item niet ook nog gebaseerd is op de laatste items van de vorige sequentie.

Een punt dat echter opvallend is, is het feit dat het netwerk er erg lang over doet om deze sequenties te leren. Het aantal benodigde trainingscycli lag rond de 60.000. Hierbij moet echter rekening worden gehouden met het gegeven dat de sequenties ook vier elementen langer zijn geworden door de toevoeging van de elementen “X”. Als wordt gekeken naar het verschil in benodigde trainingscycli tussen de eerste en de tweede simulatie, kan geconcludeerd worden dat 60.000 trainingscycli voor deze simulatie niet onverwacht is. Een vuistregel lijkt zelfs te zijn dat voor ieder item extra het netwerk 10.000 trainingscycli extra nodig heeft om de sequenties te leren. Op basis van het kleine aantal steekproeven dat hier is gebruikt, kunnen we deze vuistregel echter niet bevestigen.

Tabel 7

Output van het getrainde netwerk bij aanbieding van het tweede element van alle sequenties van simulatie 6

Input		Gegenereerde Output Trial 1						Gegenereerde Output Trial 2						Gegenereerde Output Trial 3						Gewenste Output
Draaiing	Element	Noord	Oost	Zuid	West	C	X	Noord	Oost	Zuid	West	C	X	Noord	Oost	Zuid	West	C	X	Element
L	*Oost*	*0,00*	*0,00*	*0,00*	*0,97*	*0,14*	*0,00*	*0,14*	*0,00*	*0,00*	*1,00*	*0,00*	*0,00*	*0,49*	*0,00*	*0,00*	*1,00*	*0,00*	*0,00*	*NoordWest*
R	NoordOost	0,00	1,00	1,00	0,00	0,00	0,00	0,00	1,00	1,00	0,00	0,00	0,00	0,00	0,54	1,00	0,18	0,00	0,00	ZuidOost
L	*West*	*0,99*	*1,00*	*0,01*	*0,00*	*0,00*	*0,00*	*0,04*	*0,98*	*0,96*	*0,06*	*0,00*	*0,00*	*0,02*	*1,00*	*0,06*	*0,00*	*0,00*	*0,00*	*ZuidOost*
R	ZuidWest	1,00	0,00	0,00	1,00	0,00	0,00	1,00	0,02	0,00	1,00	0,00	0,00	1,00	0,00	0,00	0,98	0,00	0,00	NoordWest
L	NoordWest	0,00	0,00	1,00	1,00	0,00	0,00	0,00	0,00	1,00	0,97	0,00	0,00	0,00	0,02	0,98	0,32	0,00	0,00	ZuidWest
R	*West*	*1,00*	*1,00*	*0,00*	*0,00*	*0,00*	*0,00*	*0,01*	*1,00*	*1,00*	*0,00*	*0,00*	*0,00*	*0,30*	*1,00*	*0,00*	*0,00*	*0,00*	*0,00*	*NoordOost*
L	ZuidOost	1,00	0,53	0,00	0,49	0,00	0,00	0,03	0,00	0,00	0,00	0,93	0,00	1,00	1,00	0,00	0,34	0,00	0,00	NoordOost
R	*Oost*	*1,00*	*0,00*	*0,13*	*1,00*	*0,00*	*0,00*	*0,01*	*0,00*	*1,00*	*1,00*	*0,00*	*0,00*	*0,00*	*0,00*	*0,92*	*1,00*	*0,00*	*0,00*	*ZuidWest*

Note. Door na te gaan of de activatie van “noord” of “zuid” desgewenst groter is én of de activatie van “oost” dan wel “west” desgewenst groter is, wordt bepaald of het gewenste derde is gegenereerd. Bij de cursief weergegeven rijen wordt nagegaan of het netwerk de sequenties juist heeft gecategoriseerd door na te gaan of de activatie van “noord” dan wel “zuid” desgewenst groter is.

4. Algemene discussie

In dit onderzoek is nagegaan hoe het categoriseren van sequenties gemodelleerd kan worden. Uit literatuuronderzoek blijkt het connectionisme een geschikte benadering om cognitie te modelleren. Dit heeft geleid tot het modelleren van het categoriseren van sequenties in een neuraal netwerk. Het neurale netwerk gebruikt voor dit onderzoek was een integratie van het *feedforward network* van Rumelhart (Rogers & McClelland, 2008) en het *simple recurrent network* van Elman (Elman, 1991). Om te testen of het netwerk inderdaad in staat was sequenties te categoriseren, moest het netwerk routes leren en categoriseren naar draairichting (linksom of rechtsom). Uit de resultaten kan geconcludeerd worden dat het netwerk inderdaad in staat blijkt sequenties te categoriseren. Bij deze conclusie moet echter rekening gehouden met het exploratieve karakter van het onderzoek.

Dit hoofdstuk vervolgt zich door eerst de resultaten van de verschillende simulaties worden te bespreken aan de hand van de richtlijnen die uit de literatuurstudie volgen. Vervolgens zullen kanttekeningen en verbeteringen van het onderzoek worden besproken wat wordt gevolgd door de implicaties van het onderzoek. Tot slot worden nog mogelijke vervolgonderzoeken besproken.

4.1. Bespreking resultaten

Om na te gaan in hoeverre het netwerk is geslaagd in het categoriseren van de sequenties, zullen de resultaten aan de hand van de twee opgestelde hypotheses worden besproken. Eerst is nagegaan of het netwerk in staat is sequenties te leren. Vervolgens wordt nagegaan of het netwerk ook in staat is deze sequenties te categoriseren. Hierbij zullen de richtlijnen voor het succesvol categoriseren en sequentieleren in acht worden genomen. Tenslotte zal besproken worden of het netwerk *graceful degradation* laat zien in overeenstemming met het *mass action principle* dat in de inleiding is besproken (Purves et al., 2008)

4.1.1. Sequentieleren

De eerste hypothese stelt dat het netwerk in staat is sequenties te leren. Deze hypothese kan aan de hand van de gevonden resultaten grotendeels worden bevestigd. Met uitzondering van simulatie 5 had het netwerk namelijk geen problemen met het leren van de sequenties. Bij simulatie 5 ondervond het netwerk problemen door het meerdere keren na elkaar herhalen van een bepaald element. Met uitzondering van sequenties die deze eigenschap hebben, kan het netwerk dus sequenties leren. Om na te gaan of het netwerk ook dezelfde eigenschappen van sequentieleren heeft als in de literatuur zijn beschreven, worden de richtlijnen uit de inleiding aangehaald.

1. *Het model toont primacy en recency effecten bij het leren van sequenties, wat betekent dat elementen aan het begin en aan het einde van de sequentie met grotere nauwkeurigheid worden geleerd dan elementen in het midden.*

Uit de resultaten valt niet af te leiden of er *primacy* en *recency* effecten optreden. De sequenties werden namelijk met een dergelijk grote nauwkeurigheid geleerd dat er geen fouten af te lezen waren. Daarnaast is dit netwerk niet geschikt om *primacy* en *recency* effecten te detecteren, omdat bij het reproduceren van de sequentie ieder element wordt voorafgegaan door een aanwijzing. Bij *serial recall* moeten alle elementen in een keer worden opgeroepen uit het geheugen zonder aanwijzingen tussendoor. Om *primacy* en *recency* effecten te ontdekken, moet het netwerk dus in staat zijn met een enkele instructie achtereenvolgens alle elementen van een sequentie te produceren.

2. *Het model leert kortere sequenties met grotere nauwkeurigheid dan langere sequenties.*

Er lijkt een invloed van sequentielengte te bestaan op het leren van de sequenties. Dit wordt geconcludeerd uit het feit dat het aantal benodigde trainingscycli om de sequenties te leren toeneemt wanneer de sequenties uit meer elementen bestaan. Om na te gaan of deze relatie ook sigmoïd is, moet meer onderzoek gedaan worden.

3. *Het model moet in staat zijn om te gaan met het voorkomen van herhaalde elementen in een sequentie zonder dat het leren van de sequenties wordt beïnvloed.*

Het netwerk lijkt tot op zekere hoogte met herhalingen om te kunnen gaan. Doordat het netwerk gebruik maakt van *compound chaining* (Henson, 1998) kan het situaties onderscheiden waarbij één item als aanwijzing dient voor verschillende items. Dit gebeurt door rekening te houden met de items die hieraan voorafgaan. Er doet zich echter een probleem voor als de voorgaande items gelijk zijn. Deze beperking van het *simple recurrent network* van Elman (Elman, 1991) doet zich voor in simulatie 5. In deze simulatie wordt een element "C" namelijk vijf keer herhaald, voordat een ander element "x" volgt. In deze situatie dient het meermaals voorkomen van het element "C" als aanwijzing voor zowel element "C" als element "x". De resultaten ondersteunen hiermee de theorie dat ook *compound chaining* geen ideale representatie van sequenties is. *Compound chaining* werkt in veel situaties, maar de grenzen van deze methode worden zichtbaar zodra een bepaald element meerdere keren na elkaar in een sequentie voorkomt.

4.1.2. Categoriseren

De tweede hypothese stelt dat het netwerk in staat is de geleerde sequenties te categoriseren. Deze hypothese kan aan de hand van de gevonden resultaten worden bevestigd. Met uitzondering van simulatie 1 en simulatie 5 was het netwerk namelijk bij iedere simulatie bij tenminste één proefneming in staat de sequenties ook te categoriseren. Het ontbreken van succes bij simulatie 1 en simulatie 5 kan door andere factoren verklaard worden. Bij simulatie 1 werd namelijk geen gebruik gemaakt van het feit dat het netwerk de verschillende elementen aan elkaar associeert, waardoor trainingsfase en testfase niet met elkaar overeen kwamen. Bij simulatie 5 functioneerde het netwerk niet naar wens, omdat het netwerk niet goed in staat is sequenties te leren waarbij een bepaald element meerdere keren na elkaar wordt herhaald. Zodra ook nog een gedeelte van de input wordt weggelaten, namelijk het

figuurnummer, is het netwerk niet meer in staat de gewenste output te genereren. Om na te gaan of het netwerk ook dezelfde eigenschappen van categoriseren heeft als in de literatuur zijn beschreven, worden de richtlijnen uit de inleiding aangehaald.

1. *Het model categoriseert via het proces van progressieve differentiatie, waarbij globale categorieën eerder geleerd worden dan specifieke categorieën.*

Of het netwerk de sequenties categoriseert volgens het proces van progressieve differentiatie van meer globale naar meer specifieke categorieën valt uit de simulaties niet af te leiden. Dit komt door het feit dat er slechts één orde categorieën werd onderzocht, namelijk de twee draairichtingen. Om te kunnen testen of het netwerk leert via progressieve differentiatie, moeten sequenties worden geleerd die in meerdere ordes te categoriseren zijn.

2. *Het model categoriseert "typische" exemplaren van een categorie sneller en met grotere nauwkeurigheid dan niet "typische" exemplaren.*

Ook de vraag of typische exemplaren eerder en nauwkeuriger worden geleerd, kon niet worden onderzocht. De reden hiervoor was dat de gebruikte sequenties slechts een enkele eigenschap hadden die hen tot de categorie linksom of rechtsom deed behoren. Het ging hier om de volgende eigenschap: "gaat na het tweede item in noordelijke dan wel zuidelijke richting". Omdat het prototype van een categorie dus slechts één eigenschap heeft, die alle leden van de categorie ook bezitten, zijn de verschillende sequenties allemaal even "typisch".

3. *Het model is in staat verschillende representaties van bepaald concept op te roepen afhankelijk van de context.*

Uit de simulaties blijkt dat het netwerk de sequenties dynamisch kan representeren. Dit blijkt uit het feit dat het netwerk in staat was de verschillende startelementen van elkaar te onderscheiden aan de hand van een gegeven context, namelijk figuurnummer en draairichting. Ook bij simulatie 6 waar het interne geheugen na iedere sequentie werd gewist, was het netwerk in staat te leren welk element moest volgen na startelement "C". Dit bevestigt de

bevindingen van Bruijnes (2011) die al eerder aantoonde dat een combinatie van het *feedforward network* van Rumelhart (Rogers & McClelland, 2008) en het *simple recurrent network* van Elman (Elman, 1991) in staat was tot een dynamische representatie van sequenties.

4.1.3. *Graceful degradation*

Uit de resultaten kan geconcludeerd worden dat het netwerk functioneert via *graceful degradation*. Wanneer bepaalde componenten van het netwerk werden uitgeschakeld, werd de functionaliteit namelijk niet heel ernstig beperkt. Dit is het beste te zien in de laatste simulatie, omdat deze het best functioneert. Deze simulatie toont dat ook wanneer het netwerk geen figuurnummer als context meekrijgt, het in staat is het juiste derde element van de sequentie te genereren na aanbieding van het eerste en het tweede element. Alleen bij sequenties die lieten zien of het netwerk de sequenties ook kan categoriseren, produceerde het netwerk niet altijd de juiste output. Bij eerdere simulaties is moeilijk iets te zeggen over *graceful degradation*, omdat uit die simulaties naar voren kwam dat het netwerk nog niet goed functioneerde en/of er met sommige relevante factoren nog geen rekening was gehouden. Een voorbeeld is dat bij simulatie 1 en 2 geen rekening is gehouden met de associaties die het netwerk legt tussen de verschillende elementen uit de sequentie. Het feit dat het netwerk functioneert via *graceful degradation* toont aan dat het in overeenstemming is met het *mass action principle* (Purves et al., 2008) wat het model van het categoriseren van sequenties biologisch geloofwaardig maakt.

4.2. Kanttekeningen en verbeteringen

Ten eerste moet bij het gebruik van dit netwerk continu rekening worden gehouden met de eigenschappen van zowel het *feedforward network* van Rumelhart (Rogers & McClelland, 2008) als het *simple recurrent network* van Elman (Elman, 1991). Bij de eerste simulatie in dit onderzoek is bij het categoriseren van de sequenties geen rekening gehouden

met het feit dat de sequenties met behulp van *chaining* zijn gerepresenteerd. Dit veroorzaakte ongewenste resultaten. Ook elementen die aan het op te roepen element voorafgaan moeten dus worden aangeboden omdat dit inherent is aan *chaining*. Bij het uitvoeren van andere taken dan het reproduceren van sequenties, moeten de sequenties dus ook met behulp van *chaining* gerepresenteerd worden.

Ten tweede was het opvallend dat bij verschillende simulaties het netwerk slechts bij één proefneming in staat was de sequenties te categoriseren. Een reden hiervoor is het feit dat de leermethode *backpropagation* gevoelig is voor de initiële staat van het netwerk (Kolen & Pollack, 1990). In dit onderzoek heeft het netwerk bij iedere proefneming een sterk verschillende initiële staat vanwege het randomiseren van de connectiegewichten voordat de sequenties worden geleerd. Doordat *backpropagation* de error verkleind door middel van *gradient descent*, wordt het leerproces gestopt als een minimum in de errorfunctie wordt bereikt. Dit kan echter een lokaal minimum zijn, wat betekent dat niet de best mogelijke oplossing is gevonden. Zodra een netwerk uit meerdere verborgen lagen neuronen bestaat, is het moeilijk om met behulp van *backpropagation* de connectiegewichten zo aan te passen dat een goede oplossing wordt gevonden (Hinton & Salakhutdinov, 2006). Een methode die hier op inspeelt is *deep learning*.

Bij *deep learning* wordt het netwerk getraind nog voordat *backpropagation* wordt toegepast (Hinton & Salakhutdinov, 2006). Met *deep learning* wordt eerst iedere verborgen laag neuronen apart getraind. Op deze manier wordt een initiële staat van het netwerk geforceerd met connectiegewichten die al dicht bij een goede oplossing ligt. *Backpropagation* dient deze gewichten dan alleen nog af te stemmen met behulp van *gradient descent*.

Tenslotte moet bij het interpreteren van de resultaten rekening worden gehouden met het feit dat dit een exploratief onderzoek is. Er is gebruik gemaakt van kleine datasets waardoor sterke conclusies uit dit onderzoek niet kunnen worden getrokken. Het heeft dan

ook weinig zin om te toetsen of de verschillende simulaties van elkaar verschillen. Wel geeft dit onderzoek een eerste indicatie dat het integreren van verschillende neurale netwerken om macrogedrag te modelleren vruchtbaar kan zijn.

4.3. Implicaties

Dit onderzoek was een eerste poging om twee neurale netwerken die microgedrag redelijk goed modelleren te integreren om zo macrogedrag te modelleren. De microgedragingen categoriseren en sequentieleren werden namelijk geïntegreerd tot het categoriseren van sequenties. Daarbij is rekening te houden met het exploratieve karakter van het onderzoek. De resultaten leveren een eerste indicatie dat zulke integraties van microgedragingen succesvol zijn. Vervolgonderzoek naar de integratie van modellen van microgedrag tot modellen van macrogedrag is daarom gewenst. Als vervolgonderzoek de stelling bevestigt dat connectionistische modellen van microgedragingen geïntegreerd kunnen worden tot modellen van macrogedragingen, levert dit bewijs voor de connectionistische benadering als succesvolle benadering voor het modelleren van cognitie. Dit levert vervolgens meer inzicht in of meer zekerheid over de werking van het brein.

Daarnaast zijn er ook praktische implicaties van het onderzoek. Het feit dat modellen die relatief eenvoudig gedrag kunnen modelleren geïntegreerd kunnen worden tot modellen die complexer gedrag kunnen modelleren, heeft praktische gevolgen. Deze modellen kunnen bijvoorbeeld worden gebruikt in robots om hen complexer “gedrag” te laten uitvoeren. Het categoriseren van sequenties zou bijvoorbeeld gebruikt kunnen worden om recepten te sorteren op basis van de handelingen die verricht moeten worden.

4.4. Vervolgonderzoek

Vervolgonderzoek kan verschillende richtingen opgaan. Een eerste vervolgonderzoek zou zich direct kunnen richten op het uitwerken van dit onderzoek. Zo zou dezelfde taak kunnen worden uitgevoerd met grotere datasets, zodat sterkere conclusies kunnen worden

getrokken. Hier zou bijvoorbeeld een clusteranalyse kunnen worden toegepast om na te gaan hoe het netwerk de sequenties groepeert. Daaraast zou het onderzoek hetzelfde netwerk een andere taak kunnen laten uitvoeren, waar meerdere ordes categorieën zijn te onderscheiden en ieder exemplaar van een categorie meerdere kerneigenschappen kan bezitten. Op deze manier kan worden nagegaan of het categorisatieproces inderdaad verloopt via progressieve differentiatie (Pauen, 2002) en of "typische" exemplaren van een categorie inderdaad sneller en nauwkeuriger worden geleerd (Posner & Keele, 1968; Rosch, 1975).

Een tweede vervolgonderzoek zou zich kunnen richten op het integreren van bestaande representaties van sequenties, zodat de nadelen van de ene representatie worden ondervangen door de voordelen van een andere. Uit dit onderzoek blijkt namelijk wederom dat het representeren van sequenties met behulp van *chaining* grenzen heeft. Andere representaties die nu beschikbaar zijn hebben echter ook zwakke punten. Dit leidt tot de vraag of er wel een representatie is die succesvol is in alle taken waarbij het leren van sequenties nodig is. Het integreren van bestaande representaties levert misschien een oplossing.

Een derde vervolgonderzoek zou zich ten slotte kunnen richten op het integreren van verschillende types neurale netwerken. In dit onderzoek is bewust gekozen om twee types *feedforward* netwerken te integreren, maar is het ook mogelijk om zowel een *feedforward* als een *recurrent* netwerk te integreren? Als wederom het categoriseren van sequenties wordt onderzocht, zou het recurrente netwerk van Botvinick & Plaut (2006) kunnen worden gebruikt om sequenties te leren. Dit netwerk kan namelijk meer benchmarkfenomenen van sequentieleren verklaren dan het netwerk van Elman kan (Botvinick & Plaut, 2006).

Referenties

Ballard, D., & Sprague, N. (2006). Modeling the brain's operating system using virtual humanoids. *International Journal of Pattern Recognition and Artificial Intelligence, 20*(6), 797-815. doi:10.1142/S0218001406004971

Bechtel, W., & Abrahamsen, A. (2002). *Connectionism and the mind: Parallel processing, dynamics, and evolution in networks.* Oxford: Blackwell Publishing.

Bjork, R. A., & Whitten, W. B. (1974). Recency-sensitive retrieval processes in long-term free recall. *Cognitive Psychology, 6*, 173-189. doi:10.1016/0010-0285(74)90009-7

Botvinick, M. M., & Plaut, D. C. (2006). Short-term memory for serial order: A recurrent neural network model. *Psychological Review, 113*(2), 201-233. doi:10.1037/0033-295X.113.2.201

Bruijnes, M. (2011). *Development of spatial mental representations in an embodied artificial neural network using sequential and egocentric data.* (Unpublished master thesis) Enschede: University of Twente.

Clegg, B. A., DiGirolamo, G. J., & Keele, S. W. (1998). Sequence learning.*Trends in Cognitive Sciences, 2*(8), 275-281. doi: 10.1016/S1364-6613(98)01202-9

Collins, A. M., & Quillian, M. R. (1969). Retrieval time from semantic memory. *Journal of Verbal Learning and Verbal Behavior, 8*(2), 240-247. doi:10.1016/S0022-5371(69)80069-1

Deese, J., & Kaufman, R. A. (1957). Serial effects in recall of unorganized and sequentially organized verbal material. *Journal of Experimental Psychology, 54*(3), 180-187. doi:10.1037/h0040536

Dinsmore, J. (1992). *Closing the gap: Symbolicism versus connectionism.* New Jersey: Lawrence Erlbaum Associates.

Elman, J. L. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning, 7*, 195-225. doi:10.1007/BF00114844

Gauthier, I., Tarr, M. J., Anderson, A. W., Skudlarski, P., & Gore, J. C. (1999). Activation of the middle fusiform 'face area' increases with expertise in recognizing novel objects. *Nature Neuroscience, 2*(6), 568-573.

Gleitman, H., Gross, J., & Reisberg, D. (2010). *Psychology.* London: Norton & Company.

GoogleDictionary (2012). Retrieved January 2013 from http://www.google.nl/webhp?sourceid=chrome-instant&ion=1&ie=UTF-8#hl=en&tbo=u&q=sequence&tbs=dfn:1&sa=X&ei=HA_qUPKAOuS80QWM1oGAAw&sqi=2&ved=0CCoQkQ4&fp=1&bpcl=40096503&ion=1&biw=1422&bih=776&bav=on.2,or.r_gc.r_pw.r_qf.&cad=b

Hecht-Nielsen, R. (1989). Theory of the backpropagation neural network. *International Joint Conference on Neural Networks*, (pp. 593-605). San Diego.

Henson, R. N. (1998). Short-term memory for serial order: The start-end model. *Cognitive Psychology, 36*(2), 73-137. doi:10.1006/cogp.1998.0685

Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 504-507. doi:10.1126/science.1127647

Jahnke, J. C. (1965). Primacy and recency effects of serial-position curves of immediate recall. *Journal of Experimental Psychology, 70*(1), 130-132. doi:10.1037/h0022013

Kolen, J. F., & Pollock, J. B. (1990). Back propagation is sensitive to initial conditions. *Complex Systems, 4*(3), 269-280.

Lee, H. (2010). Workshop on deep learning and unsupervised feature learning. *Neural Information Processing Systems.* Vancouver.

Lewandowsky, S., & Murdock, B. B. (1989). Memory for serial order. *Psychological Review, 96*(1), 25-57. doi:10.1037/0033-295X.96.1.25

Mandler, J. M. (2000). Perceptual and conceptual processes in infancy. *Journal of Cognition and Development, 1*(1), 3-36. doi:10.1207/S15327647JCD0101N_2

Marshall, P. H., & Werder, P. R. (1972). The effects of the elimination of rehearsal on primacy and recency. *Journal of Verbal Learning and Verbal Behavior, 11*(5), 649-653. doi:10.1016/S0022-5371(72)80049-5

McClelland, J. L., & Rogers, T. T. (2003). The parallel distributed processing approach to semantic cognition. *Nature Reviews Neuroscience, 4*(4), 310-322. doi:10.1038/nrn1076

McClelland, J. L., Botvinick, M. M., Noelle, D. C., Plaut, D. C., Rogers, T. T., Seidenberg, M. S., & Smith, L. B. (2010). Letting structure emerge: Connectionist and dynamical systems approaches to cognition. *Trends in Cognitive Sciences, 14*(8), 348-356. doi:10.1016/j.tics.2010.06.002

MemBrain (2010). Version 03.06.02.00, downloaded September 2012 via www.membrain-nn.de.

Mervis, C. B., & Pani, J. R. (1980). Acquisition of basic object categories. *Cognitive Psychology, 12*, 496-522. doi:10.1016/0010-0285(80)90018-3

Miller, G. A. (1956). The magical number seven plus or minus two: Some limits on our capacity for processing information. *Psychological Review, 63*(2), 81-97. doi:10.1037/h0043158

Murdock, B. B. (1962). The serial position effect of free recall. *Journal of Experimental Psychology, 64*(5), 482-488. doi:10.1037/h0045106

Page, M. P., & D, N. (1998). The primacy model: A new model of immediate serial recall. *Psychological Review, 105*(4), 761-781. doi:10.1037/0033-295X.105.4.761-781

Pauen, S. (2002). The global-to-basic level shift in infants' categorical thinking: First evidence from a longitudinal study. *International Journal of Behavioral Development, 26*(6), 492-499. doi:10.1080/01650250143000445

Posner, M. I., & Keele, S. W. (1968). On the genesis of abstract ideas. *Journal of Experimental Psychology, 77*(3), 353-363. doi:10.1037/h0025953

Purves, D., Brannon, E.M., Cabeza, R., Huettel, S.A., LaBar, K.S., Platt, M.L., & Woldorff, M.G. (2008). *Principles of cognitige neuroscience.* Sunderland: Sinauer Associates.

Rogers, T. T., & McClelland, J. L. (2008). Précis of semantic cognition: A parallel distributed processing approach. *Behavioral and Brain Sciences, 31*(6), 689-749. doi:10.1017/S0140525X0800589X

Rosch, E. (1975). Cognitive representations of semantic categories. *Journal of Experimental Psychology, 104*(3), 192-233. doi:10.1037/0096-3445.104.3.192

Schuck, N. W., Gaschler, R., Keisler, A., & Frensch, P. A. (2012). Position–item associations play a role in the acquisition of order knowledge in an implicit serial reaction time task. *Journal of Experimental Psychology: Learning, Memory, and Cognition, 38*(2), 440-456. doi:10.1037/a0025816

Smith, E. E., & Kosslyn, S. M. (2009). *Cognitive psychology: Mind and brain.* New Jersey: Pearson Education.

Van der Velde, F. (2011a). *Activation in MemBrain.* (Unpublished educational material) Enschede: University of Twente.

Van der Velde, F. (2011b). *Learning in Membrain.* (Unpublished educational material) Enschede: University of Twente.

Bijlage A. Output, horende bij simulatie 5, van het testen in hoeverre de sequenties waren geleerd toen een error van ongeveer ,006 was bereikt. De recurrente verbindingen waren niet meegeleerd.

Input			Gegenereerde output					Gewenste output
Figuur	Draaiing	Element	Noord	Oost	Zuid	West	C	Element
1	L	C	0,01	0,15	0,02	0,01	0,86	Oost
1	L	Oost	0,99	0,05	0,00	1,00	0,00	NoordWest
1	L	NoordWest	0,00	0,00	1,00	1,00	0,01	ZuidWest
1	L	ZuidWest	0,01	0,16	0,03	0,01	0,88	C
1	L	C	0,01	0,15	0,02	0,01	0,86	C
1	L	C	0,01	0,15	0,02	0,01	0,86	C
1	L	C	0,01	0,15	0,02	0,01	0,86	C
1	L	C	0,01	0,15	0,02	0,01	0,86	C
1	R	C	0,01	0,13	0,03	0,01	0,85	NoordOost
1	R	NoordOost	0,00	0,96	0,99	0,00	0,05	ZuidOost
1	R	ZuidOost	0,07	0,01	0,02	0,93	0,08	West
1	R	West	0,01	0,07	0,02	0,02	0,84	C
1	R	C	0,01	0,13	0,03	0,01	0,85	C
1	R	C	0,01	0,13	0,03	0,01	0,85	C
1	R	C	0,01	0,13	0,03	0,01	0,85	C
1	R	C	0,01	0,13	0,03	0,01	0,85	C
2	L	C	0,04	0,00	0,02	0,10	0,88	West
2	L	West	0,04	1,00	0,95	0,01	0,00	ZuidOost
2	L	ZuidOost	0,99	1,00	0,00	0,00	0,01	NoordOost
2	L	NoordOost	0,00	0,00	0,00	0,00	0,99	C
2	L	C	0,04	0,00	0,02	0,10	0,88	C

2	L	C	0,04	0,00	0,02	0,10	0,88	C
2	L	C	0,04	0,00	0,02	0,10	0,88	C
2	L	C	0,04	0,00	0,02	0,10	0,88	C
2	R	C	0,04	0,00	0,02	0,10	0,88	ZuidWest
2	R	ZuidWest	1,00	0,02	0,00	0,99	0,00	NoordWest
2	R	NoordWest	0,25	0,94	0,08	0,00	0,05	Oost
2	R	Oost	0,03	0,00	0,02	0,12	0,90	C
2	R	C	0,04	0,00	0,02	0,10	0,88	C
2	R	C	0,04	0,00	0,02	0,10	0,88	C
2	R	C	0,04	0,00	0,02	0,10	0,88	C
2	R	C	0,04	0,00	0,02	0,10	0,88	C
3	L	C	0,04	0,00	0,02	0,12	0,90	NoordWest
3	L	NoordWest	0,00	0,00	1,00	1,00	0,00	ZuidWest
3	L	ZuidWest	0,36	1,00	0,01	0,00	0,05	Oost
3	L	Oost	0,01	0,02	0,00	0,00	0,98	C
3	L	C	0,04	0,00	0,02	0,12	0,90	C
3	L	C	0,04	0,00	0,02	0,12	0,90	C
3	L	C	0,04	0,00	0,02	0,12	0,90	C
3	L	C	0,04	0,00	0,02	0,12	0,90	C
3	R	C	0,06	0,00	0,02	0,07	0,87	West
3	R	West	0,61	1,00	0,19	0,00	0,00	NoordOost
3	R	NoordOost	0,00	1,00	0,98	0,00	0,02	ZuidOost
3	R	ZuidOost	0,07	0,01	0,01	0,08	0,82	C
3	R	C	0,07	0,00	0,02	0,06	0,86	C
3	R	C	0,06	0,00	0,02	0,07	0,87	C

3	R	C	0,06	0,00	0,02	0,07	0,87	C
3	R	C	0,06	0,00	0,02	0,07	0,87	C
4	L	C	0,01	0,15	0,03	0,01	0,88	ZuidOost
4	L	ZuidOost	0,95	0,96	0,00	0,02	0,09	NoordOost
4	L	NoordOost	0,03	0,00	0,00	0,95	0,02	West
4	L	West	0,01	0,16	0,03	0,01	0,85	C
4	L	C	0,01	0,15	0,03	0,01	0,88	C
4	L	C	0,01	0,15	0,03	0,01	0,88	C
4	L	C	0,01	0,15	0,03	0,01	0,88	C
4	L	C	0,01	0,15	0,03	0,01	0,88	C
4	R	C	0,01	0,15	0,03	0,01	0,87	Oost
4	R	Oost	0,03	0,00	0,93	1,00	0,01	ZuidWest
4	R	ZuidWest	0,99	0,00	0,00	1,00	0,07	NoordWest
4	R	NoordWest	0,01	0,06	0,01	0,03	0,84	C
4	R	C	0,01	0,15	0,03	0,01	0,88	C
4	R	C	0,01	0,15	0,03	0,01	0,87	C
4	R	C	0,01	0,15	0,03	0,01	0,87	C
4	R	C	0,01	0,15	0,03	0,01	0,87	C