Kunstmatige Intelligentie 2009-2010
Oplossingen van examenvragen

Tim Besard Dimitri Roose

17 juni 2010

Hoofdstuk 1

Kunstmatige intelligentie

P6 - Bespreek de manier van leren bij leren optellen, leren lezen en leren fietsen.

Essentieel bestaan er drie verschillende manieren van leren. Vooreerst is er het **algoritmisch leren**, waarbij aangeleerd wordt hoe het oplossingsalgoritme in zijn werk gaat. Dit komt voor bij het aanleren van optellen, waarbij de leraar de methode uitlegt aan zijn leerling.

Een volgende manier is het **leren met supervisie**, waarbij het algoritme niet uitgelegd wordt, maar aan de hand van voorbeelden (een input met bijhorende output) aangeboden wordt. Zo leert men lezen, door een reeks neergeschreven letters en woorden aan de leerling te tonen, en tegelijk uit te leggen wat ze betekenen, zonder daarbij exact uit te leggen hoe het leesproces moet aangevat worden.

Tenslotte is er nog het **leren zonder supervisie**, waarbij er zich volledig zelfstandig meester moet gemaakt worden van de techniek (zonder daarbij een voorbeeldoplossing te krijgen). Dit is het geval bij leren fietsen: hoewel er een beperkte algoritmische component aanwezig is (het uitleggen voor wat de pedalen dienen, etc.) moet de leerling zelf ondervinden hoe het fietsen in zijn werk gaat.

P10 - Bereken de informatieinhoud van een tekst.

De tekst bestaat uit:

- A: 120
- B: 20
- C: 10
- D: 10

De informatieinhoud (of ook de entropie) van een bericht (in bits) wordt berekend via de formule: $-log_2(p(B))$. De term $p(B)$ is hierbij de waarschijnlijkheid dat een

bericht voorkomt (hoe lager de kans, hoe groter de informatieinhoud, wat dus de "verrassing" uitdrukt). Aangezien we het totaal aantal karakters in het bericht kennen ($120 + 20 + 10 + 10 = 160$), kunnen we kans en dus de informatieinhoud van ieder symbool berekenen. Om de informatieinhoud van de gehele tekst te berekenen, moeten we bovendien ook weten dat de informatieinhoud additief is, en we dus eenvoudig de verschillende informatieinhouden kunnen optellen:

$$\begin{aligned} ii(A) &= -log_2(120/160) = 0.42 \\ ii(B) &= -log_2(20/160) = 3 \\ ii(C) &= -log_2(10/160) = 4 \\ ii(D) &= -log_2(10/160) = 4 \\ \hline ii(tekst) &= -120 * ii(A) - 20 * ii(B) \\ &\quad -10 * ii(C) - 10 * ii(D) = 189.80 \end{aligned}$$

Men zegt ook dat de entropie de mate uitdrukt waarin informatie ontbreekt als de ordening niet gekend is.

P12 - Leg de formule uit waarmee het ID3 algoritme de informatieinhoud geleverd door een correcte klassering bepaalt.

We gaan uit van niet-ingedeelde verzameling van S elementen, die we willen indelen in k klasse, K_1, K_2, ..., K_k. Hierbij is $A(S, i)$ het aantal elementen uit S die behoort tot klasse K_i, en is het totaal aantal elementen in S, $|S|$, gelijk aan $\sum_{i=1}^{k} A(S, i)$. De informatieinhoud die een correcte klassering oplevert, berekenen we als volgt:

$$\begin{aligned} E(S) &= \sum_{i=1}^{k} A(S,i)(-log_2(\frac{A(S,i)}{|S|})) \\ &= |S|log_2(|S|) + \sum_{i=1}^{k} A(S,i)(-log_2(A(S,i))) \end{aligned}$$

Het ID3 algoritme is een inhalig algoritme, dat lokaal steeds het maximum zoekt. De bepaling van de juiste attributen gebeurt natuurlijk aan de hand van een leerverzameling L, aangezien hier de relatieve frequenties bij gekend zijn en dus de informatieinhoud kan berekend worden. De informatieinhoud die een klassering volgens het j-de attribuut (dat de leerverzameling L opdeelt in deelverzamelingen $L_{j,1}$, $L_{j,2}$, ..., L_{j,n_j}), berekent men als volgt:

$$I(j) = E(L) - \sum_{m=1}^{n_j} E(L_{j,m}) \tag{1.1}$$

Hieruit kunnen we afleiden dat de informatieinhoud van een klassering volgens attribuut j groot is als de verdeling van de verschillende klassen sterk verschilt van de verdeling in L zelf, terwijl die juist heel klein (minimaal 0) zal zijn indien de nieuwe verdeling er juist sterk op gelijkt.

De methode kent ook bepaalde **zwaktes**: zo faalt het wanneer attributen enkel gecombineerd informatie opleveren, en kan het niet detecteren als informatie uit een bepaald attribuut ook in een ander attribuut voorkomt. Hiervoor zijn een aantal verfijningen ontwikkeld:

- Drempelwaarde: indien niet overschreden, wordt de verzameling niet verder opgesplitst. Dit voorkomt een nutteloze opdeling in geval van ruis.
- Informatiewinst wordt berekend voor elk paar van attributen: dit maakt het mogelijk om combinaties te detecteren.
- Continue waarden: worden ingevoerd door een knoop met drempelwaarde in te voeren (alle items kleiner gaan links, de andere gaan rechts).

P15 - Wat is de definitie van een groep bij groeperen zonder supervisie? Waarom? Geef 2 de belangrijke criteria.

Een **groep** van punten is dat wat door een expert als een groep beschouwd wordt. Dit is geen praktische definitie, die de nadruk legt op een belangrijke eigenschap van het begrip: het is niet erg duidelijk. Ook hangt het sterk af van het toepassingsgebied.

Als praktisch alternatief stellen we 2 **criteria** voor:

1. Twee punten behoren tot dezelfde groep als ze zeer dicht bij elkaar liggen (waarbij echter een expert nodig is om de afstandsfunctie vast te leggen).
2. Deze eigenschap wordt transitief voortgezet (echter onder voorbehoud, om het effect van ruis te verminderen).

P15 - Bespreek de classificatie met k-zwaartepunten. Zijn er nog methodes om te classificeren zonder supervisie?

Hierbij probeert men een verzameling in te delen in k groepen, waarbij k gegeven is. Elke groep G_i wordt daarbij voorgesteld door haar zwaartepunt:

$$m_i = \frac{1}{n_i} \sum_{x \in G_i} x$$

n_i is hierbij het aantal punten in de groep.

De classificatie werkt als volgt:

1. Deel alle punten willekeurig in, en maak daarbij k klassen.
2. Herhaal de volgende stappen (tot er geen veranderingen meer optreden, of het maximaal aantal iteraties bereikt is):
   (a) Bereken voor alle groepen het zwaartepunt
   (b) Deel alle punten opnieuw in, door ze te overlopen en bij de groep te rekenen bij wiens zwaartepunt ze het dichtst liggen.

(c) Als een groep te weinig punten bevat, schrap ze eventueel en deel de punten opnieuw in.

Een dergelijke indeling heet men ook een "Voronoj-diagram".

De eis dat het aantal groepen op voorhand gegeven is, vormt een beperking: geeft men een te groot aantal op dan zullen samenhangende groepen gesplitst worden, maar indien men een te klein aantal opgeeft zullen verschillende groepen worden samengenomen. Dit probleem wordt opgelost door het algoritme verschillende keren los te laten, steeds met een andere begintoestand, en er vanuit te gaan dat de ingedeelde groepen identiek moeten zijn indien het juiste aantal groepen gespecificeerd is. Een ander probleem zijn convex groepen: deze kunnen niet ingedeeld worden door gebruik te maken van deze methode.

We kunnen dergelijke classificatie ook bereiken door gebruik te maken van een Hopfield-net.

Hoofdstuk 2

Zoeken in zoekruimten

P23 - Bewijs dat een A^*-heuristiek toelaatbaar is.

Een heuristiek is A^* als voor elke knoop geldt dat $h(k) \leq h^*(k)$ (de heuristische kost mag dus nooit overschat zijn). Daarnaast is een heuristiek toelaatbaar als hij een pad vindt en als dat pad naar het doel optimaal is.

Bij het ontwikkelen van knopen volgens een A^*-heuristiek h kunnen we steeds een knoop k' vinden, waarvoor geldt:

1. k' zit in de NOK.
2. k' ligt op een optimaal pad van s naar D.
3. De schatting $g(k')$ is correct (of: $g(k') = g^*(k')$)

In de beginsituatie is logischerwijs aan deze 3 voorwaarden voldaan. Bij het verwijderen van de startknoop uit de NOK, wordt zijn plaats vervangen door zijn opvolger k', waarvoor de drie voorwaarden gelden:

1. De opvolger is een kind van de startknoop, en zit dus in de NOK.
2. ??? (pagina 23)
3. De gekende kost kan berekend worden door de gekende kost van de voorganger te nemen, met daarbij de kost om naar de opvolger te gaan, wat (aangezien de knoop op het optimale pad ligt) steeds overeenkomt met ge reële optimale kost.

P27 - Leg uit: $\alpha - \beta$ snoeien. Hoe kan je de performantie hierbij verbeteren?

Bij eenvoudige **minimax**-evaluatie zal men, naargelang een maximaal of minimaal eindresultaat gewenst is, de knoop kiezen waarbij het minimum maximaal respectievelijk het maximum minimaal is, om in geval van een intelligente tegenspeler het beste eindresultaat te bekomen. Hierbij moet echter steeds de hele boom geëvalueerd worden, wat al snel kostelijk wordt.

Door gebruik te maken van $\alpha - \beta$ **snoeien** kunnen we het doorzoeken van bepaalde deelbomen kunnen elimineren, als blijkt dat die suboptimaal zijn (geen deel zijn van het optimale pad). We weten echter nooit wat het optimale pad is, maar we kunnen er wel grenzen aan stellen die verbeteren naargelang we de boom doorzoeken. Bij het verwerken van de boom houden we steeds twee grenzen bij:

- α: de minimumscore van de maxspeler.
- β: de maximumscore van de minspeler.

Bij het verwerken van de boom kunnen we deze twee waarden dan gebruiken om te bepalen of we een deelboom moeten verwerken:

- Als k een minknoop is, moet de winst groter zijn dan α.
- Als k een maxknoop is, moet de winst kleiner zijn dan β.

Het is dus duidelijk dat de efficiëntie van $\alpha - \beta$ snoeien gevoelig is aan de volgorde waarin we de kinderen evalueren: als de eerst gecontroleerde deelboom een goede zet is, worden de grenzen scherp gesteld waardoor veel andere bomen zullen kunnen geëlimineerd worden.

De methode kan eventueel verbeterd worden door gebruik te maken van verschillende technieken:

- Statische evaluatie: de boom slechts evalueren tot op een zekere vooraf-ingestelde diepte.
- Dynamische evaluatie: bepalen van de evaluatie van de andere knopen door minimax met snoeien.

Hoofdstuk 3

Expertsystemen

P31 - Leg grondig uit: voorwaarts en achterwaarts aaneenschakelen. Hoe wordt dit toegepast bij een eenvoudig expertsysteem?

Voorwaarts aaneenschakelen worden gebruikt bij het efficiënt zoeken van een oplossing in een regelbank. De normale verwerkingsmethode, **voorwaarts redeneren**, baant zich sequentieel een weg doorheen de regelbank, en introduceert steeds meer gegevens (waarbij steeds de regel in kwestie uit de regelbank verwijderd wordt) tot een deel van de doeltoestand bereikt is. Aangezien hierbij veel regels steeds opnieuw gecontroleerd moeten worden, introduceert men **voorwaarts aaneenschakelen**. Deze methode introduceert twee extra datastructuren: voor elke regel een premisseteller, en een lijst met regels die voor elke uitspraak bijhoudt bij welke regels ze in de premisse voorkomt. Nu kan bij het evalueren eenvoudig de premisseteller van relevante regels verlaagd worden, waarna de conclusie van regels met premisseteller op 0 aan de gegevens kunnen toegevoegd worden.

Aangezien bij voorwaartse methoden veel regels geëvalueerd worden die niet bijdragen tot het doel, gebruikt men soms **achterwaarts redeneren**. Deze complexere methode begint bij het doel, en zoekt zo naar regels (eventueel stapsgewijs, door een lijst met tussendoelen bij te houden) die de premisse van het doel realiseren. Omdat hierbij opnieuw steeds alle regels moeten overlopen worden, kan men dit analoog aan voorwaarts aaneenschakelen, optimaliseren via **achterwaarts aaneenschakelen**.

P35 - Leg uit (aan de hand van gegeven formules): zekerheidsfactoren. Wat als 1 van beide gelijk is aan -1, 0 of 1?

Zekerheidsfactoren vormen een methode om onzekerheden van regels uit te drukken. Het verbetert Bayesiaans redeneren, door eenduidig te definiëren wat moet gebeuren bij een meervoudige premisse (minimum nemen), en laat experts toe om intuïtief onzekerheden uit te drukken.

Wanneer eenzelfde conclusie bepaald wordt door meerdere regels, elk met hun eigen onzekerheid, moet een juiste **combinatiefunctie** gebruikt worden naargelang de situatie:

- Beide waarden positief: aangezien de uitspraak bevestigd wordt door twee regels, moet het resultaat groter zijn dan beide waarden apart (met als bovenlimiet 1).
- Beide waarden negatief: hierbij moet analoog aan het vorige geval, het resultaat zeker kleiner zijn dan beide waarden apart (met als onderlimiet -1).
- Tegengesteld teken: in dit geval spreken de beide regels elkaar tegen. In het geval dat de waarden absoluut ongeveer even groot zijn, moet het resultaat tegen de 0 aanwegen. Wanneer echter de ene regel vrij sterk is, en de andere zwak, moet het resultaat nog altijd sterk blijven!

Wanneer echter 1 van de waarden 1 of -1 is, kan de regel in kwestie direct aangenomen worden, zonder de andere onzekerheden te moeten geëvalueerd hebben.

P38 - Wat is een semantisch net? Hoe drukt men overerving hierin uit? Wat zijn frames?

Een **semantisch net** is een model voor de opslag van de betekenis van woorden in het menselijke brein. Het bestaat uit een geëtiketteerde graaf, waarbij elke knoop een *begrip* is, verbonden met andere begrippen via een gelabelde verbinding. Sommige van die verbindingen drukken daarbij **overerving** uit, door middel van een "is een" relatie. Bij het zoeken van een antwoord op een vraag kan, na het overlopen van eventuele andere verbindingen, een "is een" relatie naar een meer algemeen begrip gevolgd worden om vervolgens de vraag opnieuw te proberen te beantwoorden.

Een **frame**, ontwikkeld in dezelfde periode van het semantisch net, wordt gebruikt om structuur te brengen in gegevens (vergelijkbaar met klassen). Het is een hiërarchisch systeem, gemodelleerd als een boom bestaande uit frames.

Een frame heeft ook **slots** (vergelijkbaar met attributen), die een waarde kunnen bevatten, maar ook kunnen verwijzen naar andere frames. Slots hebben altijd een waarde, hetzij niet_ingevuld, en erven slots alsook de eventuele waarde over van hun ouder. Los van de eventuele ouder kan een frame nieuwe slots definiëren, en die een bijhorende waarde geven.

P41 - Leg uit: predikatenlogica. Wat zijn vrije variabelen? Hoe maak je een instantie? Wat is het nut daarvan bij regelbanken?

Door de introductie van slots die overgeërfd worden, kunnen we niet eenduidig meer verwijzen naar een bepaald gegeven van een frame. Om hiervoor toch regels te kunnen opstellen, definiëren we enerzijds een frame.slot syntax, en introduceren we anderzijds het gebruik van **predikatenlogica**. Hiermee kunnen we eenvoudig verwijzen naar een onbekend frame, waarbij een slot een bepaalde waarde aanneemt. Een dergelijke onbekende noemen we een **vrije variabele**, en we definiëren de graad van een predicaat als het aantal vrije variabelen daarin.

Als we bij een predicaat met graad > 0 vrije variabelen invullen met de namen van een frame, hebben we een **instantie** van het predicaat. We kunnen de graad echter ook

verlagen door een vrije variabele te **binden** met een kwantor, zoals "er is een" of "voor alle".

Het nut hiervan bij **regelbanken** is dat men flexibele regels kan opstellen die een relatie tussen frames uitdrukt aan de hand van de waarde van hun sloten. We doen dit door een premisse en conclusie te nemen met dezelfde graad en identiek dezelfde vrije variabelen. Als er een set frames bestaat waarvoor de premisse geldt, wordt een instantie van de conclusie toegevoegd aan de verzameling gegevens van de regelbank. Kwantoren kunnen hierbij nuttig zijn om een tussenrelatie uit te drukken die uiteindelijk niet in de conclusie voorkomt (bijvoorbeeld: een premisse "x is kind van z en y is ouder van z" met als conclusie "y is grootouder van x"). Bij het controleren van een set regels (bijvoorbeeld bij voorwaarts redeneren) zal nu ook moeten gekeken worden of er geen instantie van een regelschema bestaat dat voldoet aan de beschikbare gegevens. Wanneer een bepaald algoritme echter een regel verwijdert (bijvoorbeeld bij achterwaarts aaneenschakelen) zal nu wel enkel de instantie mogen verwijderd worden, en niet het regelschema zelf).

Om predicaten te kunnen toepassen op de overerving van framesystemen, is meestal voorzien in een aantal vaste predicaten, zoals "?x is_direct_een ?y", "?x is_een ?y" en "?x is_een_object" (en geen "klasse").

Een belangrijke toepassing van een dergelijke framesysteem met predicaten vindt men bij databases met intelligente queries.

P44 - Wat zijn facetten van slots? Verduidelijk N- en Z-overerving.

Normaal is het onmogelijk om operaties binnen een frame te plaatsen, maar binnen sommige systemen kunnen we dit doen door een **facet** aan een slot te hechten. Er bestaan verschillende soorten facetten:

- Waardefacet: een facet dat expliciet een waarde bepaalt.
- Defaultfacet: een facet die een waarde specificeert die enkel gebruikt wordt indien geen enkele andere waarde is ingevuld.
- Toekenfacet: dit facet bepaalt een waarde wanneer het slot nog geen ingevulde waarde heeft (meestal enkel wanneer benodigd). Dit wordt veel gebruikt bij terugredeneren.
- Veranderfacet: dit facet bepaalt bij het invullen of wijzigen van een slot, welke andere slots (van hetzelfde of een ander frame) ook moeten veranderd worden. Dit wordt veel gebruikt bij voorwaarts redeneren. Een dergelijk facet houdt dus enkel rekening met de inhoud van het waardefacet, en zal niet geactiveerd worden in geval van evaluatie.
- Vraagfacet: hiermee wordt extra informatie aan de gebruiker gevraagd, indien relevant.

Wanneer we een slot nodig hebben dat nog niet ingevuld is, zijn er verschillende mogelijkheden om zijn waarde te bepalen. Bij **horizontale evaluatie** wordt enkel

gekeken naar de lokale facetten, zonder daarbij aandacht te besteden aan de hiërarchie. In het geval van **verticale evaluatie** echter wordt ook gekeken naar de ouder, indien relevant. Hierbij zijn er echter twee mogelijkheden:

- N-overerving: hierbij wordt eerst de hiërarchie gerespecteerd, vooraleer naar een volgend facet te kijken.
- Z-overerving: als we deze strategie toepassen, worden eerst alle lokale facetten geëvalueerd, vooraleer te stijgen in de hiërarchie.

Hoofdstuk 4

Acties en genetische algoritmen

P48 - Wat zijn strategieën? Hoe worden ze gebruikt bij genetische algoritmen? Wat is een *default regel*, en hoe los je conflicten op in een strategie?

Een **strategie** is een geheel aan regels dat voor elke mogelijke visie een actie teruggeeft. Het beschrijft dus het gedrag van een entiteit binnen een bepaalde wereld. Om hierbij te vermijden dat voor elke mogelijke visie een regel moet geschreven worden, introduceert men **wildcards** (die een bepaalde variabele toelaten een bepaalde strekking of zelfs volledige willekeurige waarde aan te nemen), alsook een **default regel**. Regels krijgen daarbij een prioriteit (mogelijk impliciet bepaald gebaseerd op hoe specifiek ze zijn), waardoor **conflicten** eenvoudig kunnen opgelost worden. De default regel heeft daarbij natuurlijk de laagste prioriteit mogelijk.

P49 - Hoe maak je een nieuwe generatie van strategieën aan?

Selectie Eerst wordt een selectie gemaakt, waarbij enkel de beste strategieën uit de populatie worden overgehouden. Hiervoor gebruikt men een geschiktheidsfunctie, die aangeeft hoe goed de strategie presteert.

Aanvullen Om de leemtes in de populatie weer op te vullen, moeten nieuwe strategieën gegenereerd worden. Dit kan op twee manieren gebeuren, eventueel gecombineerd:

1. Kruising: hierbij worden twee strategieën uit de populatie vermengd, zodat een nakomeling ontstaat met eigenschappen van beide ouders. Om hierbij te vermijden dat een strategie gevuld geraakt met regels die sterk overlappen, is het belangrijk om op gezette momenten de regels met de grootste overlapping te verwijderen (of er maar 1 van over te houden).

2. Mutatie: dit proces neemt een strategie uit de populatie, en voert er kleine wijzigingen bij door. Meestal wordt dit eenmaal toegepast, en maar op een enkele regel.

P50 - Leg het uurrooster probleem uit. Wat zijn harde en zachte eisen? Wat moet je doen als het moeilijk is om een beginpopulatie met juiste oplossingen op te stellen?

Bij het **uurroosterprobleem** poogt men een uurrooster op te stellen, waarin een aantal studentengroepen, die elk een aantal cursussen volgen gegeven door bepaalde studenten, klaslokalen toegewezen worden op gezette uren. Dit moet aan een aantal eisen voldoen:

- **Harde eisen**: aan deze eisen moet voldaan worden, zoals het feit dat elk vak moet gegeven worden, of dat docenten maar 1 les tegelijk kunnen geven.
- **Zachte eisen**: dergelijke eisen maken het onderscheid tussen een goed en slecht lessenrooster, maar zijn niet expliciet vereist (zoals het afwezig zijn van springuren).

Als men direct alle oplossingen waarbij aan een harde eis niet voldaan is, zou verwijderen, is het quasi onmogelijk om een **beginpopulatie** van uurroosters aan te maken. Daarom is het beter om bij het falen van een harde eis een sterk negatief punt toe te kennen, in plaats van de strategie direct weg te gooien. Zo kan er op den duur geëvolueerd worden naar een beter lessenrooster, waarin aan alle harde eisen voldaan is, maar er nog verbetering mogelijk is op vlak van zachte eisen.

Hoofdstuk 5

De regel van Bayes en Markovketens

P54 - Wat zijn Markov-ketens? Leg het Viterbi-algoritme uit met betrekking tot spraakherkenning.

Een **Markov-keten** is een speciaal soort automaat, veel gebruikt bij het modelleren van tijdsafhankelijke processen, en verschilt op twee vlakken van klassieke automaten:

1. Producerende eenheid: een Markov-keten genereert uitvoer zonder uit te gaan van een specifieke invoer, en werkt dus autonoom.

2. Stochastische eenheid: de uitvoer is afhankelijk van bepaalde toevals-variabelen, zelfs al is de huidige staat gekend kan men niet eenduidig bepalen wat de volgende staat zal zijn (een Markov-keten werkt dus niet-deterministisch).

De keten bevindt zich altijd in een bepaalde staat, en de volgende staat wordt bepaald aan de hand van de probabiliteiten in de transitiematrix. Deze is enkel afhankelijk van de huidige staat, en is onafhankelijk van de tijd.

Indien er ook een probabiliteitenmatrix bestaat voor het bepalen van de uitvoer, dan spreken we van een **Hidden Markov Model** (HMM). Hierbij bestaat er niet langer een 1-op-1 relatie tussen de staat en de uitvoer, waardoor we niet langer eenduidig de staat kunnen bepalen (vandaar dat het model *verborgen* is). Bij de toepassing van HMM's identificeren we 3 mogelijke kernproblemen:

- Gegeven de parameters van het model, moet berekend worden wat de probabiliteit van een bepaalde uitvoersequentie is.

- Gegeven een bepaalde uitvoer, moet berekend worden welke statensequentie verantwoordelijk was voor het genereren van die uitvoer (vb. spraakherkenning, Viterbi-algoritme).

- Gegeven set aan outputsequenties, bereken de transitiematrix en outputprobabiliteiten.

Nu we een HMM voor de verschillende fomenen hebben, kunnen we die aaneenschakelen zodat we een HMM krijgen voor elk woord. Een volgende stap is een **grammaticaal model** opstellen, waarbij via grammaticale analyse uitgedrukt wordt hoe woorden zich verhouden tot elkaar. De finale HMM is een samensmelting van al die probabiliteiten met de verschillende woord-HMM's.

Gegeven een bepaalde uitvoer, kunnen we vervolgens het optimale pad berekenen (dit is de optimale statensequentie die de HMM uitgevoerd heeft). Dit kunnen we realiseren met het Viterbi-algoritme, waarna we weten wat de meest waarschijnlijke sequentie is die de gegeven uitvoer geproduceerd heeft.

Bespreek de tijd bij het overgaan van fonemen naar woorden.

Bij het verwerken van een gesproken tekst, samplen we in frames van ongeveer 10 ms. Die set aan frames onderwerpen we aan een HMM, waarbij we proberen bepalen welke beginstaat afhankelijk was voor het genereren van de gegeven data. Elke HMM verantwoordelijk voor een foneem, bestaat daarbij uit drie staten: Begin, Midden en Eind, gevolgd door een eindstaat. Elke staat komt overeen met een bepaalde klank, waarbij natuurlijk de staat verschillende mogelijke varianten van die klank bevat (daar het een HMM is).

P60 - Wat moet men doen om spraakherkenning aan te passen aan een bepaalde persoon?

Hiervoor laten we de persoon in kwestie een bepaalde tekst lezen, die we op voorhand kennen. Aangezien we de tekst kennen, weten we welke staten moeten doorlopen worden om de uitvoer te produceren. Mits synchronisatie tussen de staten en het ingesproken geluid kunnen we het de uitvoer- en transitieprobabiliteiten optimaliseren.

Dit biedt de volgende opportuniteiten: gebaseerd op de invoergegevens kunnen we de uitvoermatrix optimaliseren zodat die bij elk foneem van een bepaald woord overeen komt met de verschillende klanken die de persoon produceert. Ook kunnen we timing verbeteren door de transitiematrix van de fomenen-HMM te optimaliseren. Beide stappen leiden uiteindelijk tot een gepersonaliseerde data die de herkenning verbetert.

Hoofdstuk 6

Neurale netten

P63 - Bespreek de verschillen tussen een klassieke computer en een neuraal netwerk, zowel qua snelheid als qua gevoeligheid voor fouten.

We kunnen het verschil aantonen aan de hand van de essentiële karakteristieken van een klassieke computer:

- Bij een klassiek computersysteem moet men de machine tot de laaste detail vertellen wat het algoritme is om de invoer te verwerken. Het **algoritme** moet dus eigenlijk al gekend zijn voor dat men er informatie tracht mee te verwerken.
- De computer is zeer gevoelig voor **onverwachte invoer**.
- **Onregelmatigheden** in programma of apparatuur kan het volledige systeem onbruikbaar maken.
- Elk begrip en object van de verwerking is duidelijk **localiseerbaar**.

Het eerste punt is het belangrijkste verschilpunt. We kennen voor sommige taken namelijk het algoritme niet. Dit is bijvoorbeeld het geval bij lezen. We hebben dit **geleerd**, zelfs om handgeschreven letters te herkennen ookal zijn ze telkens verschillend. Dit illustreert tevens ook het tweede punt. We kunnen volgende karakteristieken opsommen voor neurale netwerken:

- Een neuraal netwerk wordt niet geprogrammeerd maar **leert** zonder een expliciet algoritme. Er hoeft trouwens niet expliciet een leerfase te zijn, het netwerk kan ook leren terwijl het informatie verwerkt.
- Het neuraal netwerk is zeer goed in **veralgemenen**.
- Het is bovendien zeer ongevoelig voor **beschadigingen**. Bij gedeeltelijke vernietiging kan het meestal nog vrij goed zijn taak uitvoeren.
- Begrippen en objecten zijn **niet exact te lokaliseren**. Het zijn combinaties van parameterwaarden die een betekenis hebben.

P66 - Leg de volgende termen uit: axon, dendriet, synaps, excitatorische en inhibitorische pulsen. Hoe werkt een biologisch net?

Een biologisch net bestaat uit **neuronen** die de basis bouwstenen vormen. De neuronen bestaan uit een centraal deel, het soma van waaruit enkele slierten vertrekken.

Eerst zijn er de lange slierten met vertakkingen, dit zijn de **dendrieten** die dienst doen als receptoren van signalen van andere neuronen.

Daarnaast heb je vertrekkend uit het soma nog een sliert, het **axon**. Ook het axon kan zeer lang zijn alsook vertakt het zich maar minder opvallend dan de dendrieten. Via het axon zendt een neuron zijn signalen uit.

De **synapsen** stellen de verbindingspunten voor tussen twee neuronen. Ze bevinden zich op het uiteinde van de vertakkingen. Een synaps heeft dus aan de ene kant een axon en aan de andere kant een dendriet. Ze brengen de pulsen van het ene neuron over naar het andere neuron. Sommige synapsen hebben een betere doorlaatbaarheid dan andere. Bovendien kan dit op langere termijn wijzigen waardoor men in staat stelt een neuraal netwerk te doen leren.

De neuronen sturen signalen door in de vorm van **pulsen** die eigenlijk elektrische stroomstoten zijn. Elk neuron kan op een gegeven moment besluiten een impuls af te vuren. De uitvoer van een neuron is een analoog signaal. De sterkte van dit signaal wordt bepaald door de tijdsduur die verloopt tussen verschillende inkomende pulsen. Een puls doorloopt het hele axon en komt dan bij de synapsen van het axon. All neuronen waarvan een dendriet verbonden is met dit axon ontvangen een de puls van het afvurende neuron.

Op basis van wat vuurt het neuron nu een puls af? We kunnen het soma opvatten als een reservoir. Naargelang het neuron waaruit de puls vertrekt zijn er twee mogelijkheden.

- een excitatorische puls vult dit reservoir met een hoeveelheid evenredig met zijn sterkte. Onder andere bepaald door doorlaatbaarheid synaps.
- een inhibitorische puls vermindert de inhoud van het reservoir evenredig met zijn sterkte.

Het soma blijft de binnenkomende pulsen echter niet vasthouden en vergeet ze dus geleidelijk aan. Als het niveau een bepaalde grenswaarde bereikt vuurt het desbetreffende neuron een puls af waardoor het reservoir wordt geledigd.

Hoofdstuk 7

Kunstmatige netwerken

P72 - Bespreek de drie manieren om tijd te modelleren bij neuronen.

Afhankelijk van de toepassing kan er ook een verschil zijn hoe de factor **tijd** verwerkt wordt in het netwerk. Vooral dan wanneer er sprake is van terugkoppeling. De verschillende methodes leveren verschillende resultaten op.

- Invoeren van een **discrete tijd**, zoals bij de computer. Elk neuron heeft op toestand t_i de uitvoer die overeenkomt met zijn ivoer op t_{i-1}.
- Een andere methode is dat neuronen alleen veranderen als we ze **aanduiden**. De vorige methode is een speciaal geval waarbij me op elk tijdstip alle neuronen aanduidt. Deze methode is iets sneller en vraagt minder geheugen dan de vorige.
- Integenstelling tot vorige methoden die zowel voor TLU's en analoge neuronen werken kunnen we bij analoge neuronen gebruik maken van een **continue overgang**. Is de invoer van een neuron x_i op ogenblik t gelijk aan

$$s_i = \sum_{j=1}^{n} w_{ji} u_j$$

dan kan de verandering van uitvoer gemodelleerd worden met een interne toestand a_i die afhangt van de tijd. De uitvoer wordt dan gegeven door

$$u_i(t) = f(a_i(t))$$

$$\frac{da_i}{dt} = -a_i(t) + s_i(t)$$

met f de excitatiefunctie. De tweede vergelijking zegt dat a_i kleiner wordt als $a_i > s_i$, evenredig met het verschil. Als $a_i < s_i$ wordt a_i groter.

P75 - Leg uit: de regel van Hebb. Hoe leert een neuraalnetwerk? Hoe verander je de threshold bij het leren?

De **regel van Hebb** is een methode om de gewichten te veranderen tijdens de leerfase van een neuraal netwerk. Elk in- en uitvoergegeven wordt gepresenteerd aan het net en de gewichten worden "in de goede richting" aangepast. Gewichten tussen twee neuronen worden vergroot in directe functie van de activiteit van de twee neuronen. Deze manier sluit zeer dicht aan bij het biologische leerproces.

Men kan het leren voor massief lerende programma's zoals artificiële neurale netten opdelen via de manier waarop het net het leeraanbod verwerkt. In tegenstelling tot biologische netten, waar er geen onderscheid is tussen gebruiken en leren, hebben de meeste artificiële netten een leer- en gebruiksfase. We gaan hier uit van een leerproces met supervisie. We proberen de gewichten op zodanige manier te passen dat de gegeven invoer de gevraagde uitvoer geeft. We onderscheiden volgende methoden:

- De **regel van Hebb** toepassen.
- Via een **graduele aanpassing** die de volgende regel niet volgt maar afgeleid is uit kennis van het probleem. Dit wordt o.a. bij systemen van stuenvectoren toegepast.
- Soms worden de gewichten bepaald uit **vergelijkingen** die een optimale waarde voor de gewichten geven. Deze manier staat het verste af van biologisch leren.

P74 - Bespreek de regel van Hebb. Van waar is die afkomstig? Leg het biologisch netwerk uit. Wat was het beroep van Hebb?

De **regel van Hebb** is een methode op neurale netwerken te leren, die dicht aanleunt bij net biologische leerproces. Daarbij worden de gewichten tussen twee neuronen vergroot in directe functie van de activiteit ervan. Hoewel we ze praktisch enkel zullen toepassen wanneer een netwerk verkeerd voorspelt, is de coëfficiënt waarmee we het gewicht in kwestie aanpassen conform de regel van Hebb direct gerelateerd aan de activiteit van dat neuron.

De **afkomst** van de regel is duidelijk: biologische neurale netwerken. Synapsen, de biologische eenheden die instaan voor het transport van signalen van een axon naar een dendriet, zullen naargelang er meer activiteit plaatsvindt tussen de twee neuronen meer of minder doorlaatbaar worden. Hierbij is de doorlaatbaarheid dus direct gerelateerd tot de activiteit tussen de neuronen.

Donald Hebb tenslotte was een Canadese psycholoog met sterke interesse op vlak van neuropsychologie. Voor een gedetailleerdere biografie verwijzen we de lezer naar de relevante Wikipedia pagina door.

Hoofdstuk 8

Informatieverwerking met neurale netten

P77 - Hoe maakt men een netwerk dat $F : \{0,1\}^n \rightarrow \{0,1\}^k$ verwezenlijkt?

Eerst en vooral proberen we een netwerk te maken voor de binaire functie $F : \{0,1\}^n \rightarrow \{0,1\}$. In geval van lineair scheidbare punten, kunnen we dit eenvoudig doen met een eenlagig netwerk (zie later). Om een willekeurige functie te verwezenlijken, kunnen we echter steeds gebruik maken van een tweelagig netwerk. Hiertoe definiëren we eerst 2 **elementaire functies**, beide te realiseren met een eenlagig net:

- De or-functie: deze functie kan men realiseren met een neuron met als drempelwaarde 0.5 en n ingangen met gewicht 1.
- De selectiefunctie: deze functie, die een enkel specifiek k-bits patroon afbeeld op 1, kunnen we maken door bij een neuron met drempelwaarde $k - 0.5$, elke verbinding een gewicht van $2b_i - 1$ te geven.

Met behulp van deze functies kunnen we nu steeds de **disjunctieve normaalvorm** realiseren, die ons in staat stelt elke logische functie te realiseren.

Vervolgens moeten we nog aantonen dat indien we een netwerk kunnen realiseren voor $k = 1$, dit ook mogelijk is voor **alle k-waarden**. Dit is eenvoudig aan te tonen: de aparte tweelaags-netwerken kunnen immers steeds samengevoegd worden tot een tweelagig-geheel.

De laatste stap is zorgen dat we het netwerk kunnen doen **leren**. Bij een eenlagig netwerk kunnen we eenvoudig de regel van Hebb toepassen, die de gewichten van het uitvoerneuron aanpast indien de uitvoer daarvan verkeerd was. Hierbij zijn meestal een aantal herhalingen nodig, daar Hebb niet garandeert dat een neuron na een enkele iteratie juist geklasseerd wordt (alsook kunnen andere neuronen plots verkeerd geklasseerd worden), waarbij men moet stoppen na een fix aantal iteraties, of indien er een stabiele foutloze toestand bereikt is. Bij tweelagige netwerken moet een complexere methode toegepast worden, “error backpropagation” genaamd.

P81 - Hoe realiseer je een automaat met behulp van een neuraal netwerk? Hoe worden staten en hun overgangen hierbij voorgesteld?

De automaat die we zullen realiseren, is een **Moore-automaat**. Deze kent de volgende eigenschappen:

- Eindige verzameling staten Q, waarbij de automaat zich steeds in een enkele staat bevindt (deterministische automaat).
- Invoeralfabet S, die alle mogelijke invoeren bevat.
- Uitvoeralfabet G, die alle mogelijke uitvoeren bevat.
- Transitiefunctie d, die aangeeft naar welke staat uit S de automaat gaat als ze een symbool uit het invoeralfabet S binnenkrijgt.
- Uitvoerfunctie l, die bepaalt welke uitvoer uit het uitvoeralfabet G de automaat genereert als ze toekomt in een staat uit Q.
- Beginstaat q.

Een dergelijke automaat heeft vermits het eindig aantal staten, ook een eindig geheugen: de automaat herinnert zich iets van invoer uit het verleden, maar niet alles.

Om deze automaat te realiseren met een **neuraal model**, maken we gebruik van tijdsafhankelijke neuronen. De uitvoer van dergelijke neuronen op het moment t hangt af van totale invoer op moment $t-1$. Hierbij wordt een gegeven invoer ook verwerkt door 2 opeenvolgende neuronen, en wordt de uitvoer pas aangevoerd op even momenten: dit om de automaat de kans te geven om voor de verwerking te zorgen. We stellen ook dat zowel de in- als uitvoer moet opgemaakt zijn als een binaire string, met slechts een enkele bit op 1. Dit is echter niet echt een beperking, daar we aan de hand van een extra laag die de selectiefunctie implementeren, steeds een vertaalfunctie kunnen maken die elke combinatie converteert naar een dergelijke string.

Om de **transitiefunctie** te realiseren, gebruiken we $n * |Q|$ neuronen die elk een EN-functie met twee ingangen implementeren: 1 ingang verbonden met een invoerneuron (er zijn altijd maar 2 invoerneuronen, 0 en 1, daar de automaat per symbool inleest)

, en 1 met de uitgang van een bepaalde staat. De uitgang wordt verbonden met het volgende neuron. Het gewicht van deze verbindingen is, net als alle andere verbindingen, gelijk aan 1. De drempel van dergelijke transitieneuronen is gelijk aan $1,5$ (EN-functie), terwijl die van statenneuronen gelijk is aan $0,5$ (OF-functie). Op elk ogenblik heeft ofwel juist 1 statenneuron de uitvoer 1 (even tijdsstippen, waarbij de transitieneuronen allemaal 0 geven), of staat exact 1 transitieneuron op 1 (oneven tijdsstippen, waarbij logischerwijs de statenneuronen allemaal 0 geven). Dit is waarom de automaat steeds 2 tijdseenheden nodig heeft om invoer te verwerken. De statenneuronen kunnen echter weggelaten worden, ze geven enkel invoer door en waren enkel ingevoerd omwille van de duidelijkheid.

Hoofdstuk 9

Het classificatieprobleem

We weten dat $F : \{0,1\}^n \rightarrow \{0,1\}^k$ kan worden verwezenlijkt met een neuraal netwerk van ten hoogste 2 lagen. Toon nu aan dat er een drielagig netwerk bestaat dat de scheiding tussen $\mathcal{L}^+$ en $\mathcal{L}^-$ geeft als deze geen enkel element gemeenschappelijk hebben.

De deltaregel (formule gegeven): leg elk element van de formule uit (de formule is gegeven). Maak een tekening. Geef ook het bewijs. Geef het verband tussen de 2 delen. Waarom is tweede versie praktischer bruikbaar?

Hoe maak je een drielagig net om punten in 2 groepen te classificeren (geef ook een tekening + duid aan hoeveel selectieneuronen er voor nodig zijn)?

Hoofdstuk 10

Steunvectoren

Bespreek de stelling van het middelloodvlak. Bewijs dat het optimaal is.

Bewijs dat H het optimale hypervlak is indien x en y respectievelijk in de convex omhullenden liggen van $\mathcal{L}^+$ en $\mathcal{L}^-$.

Wat is een kwadratische scheiding? Hoe verwezenlijk je dit in één neuron?

Gegeven de formule van de partieel afgeleide van de energie naar αk. Gevraagd is het leeralgoritme op te stellen en aan te geven waarom dit altijd een (zo goed mogelijke) oplossing geeft.

Bespreek niet-lineaire classificatie met SVM's.

Leg de energiefunctie van een SVM uit (wordt gegeven), en bewijs dat ze klopt. Waarvoor dient ze? Bespreek alle componenten.

Hoofdstuk 11

Associatieve geheugens

P104 - Wat is een Hopfield-net? Bespreek de energiefunctie ervan (wat ze doet, en wat er gebeurd als neuronen omkippen). Hoe leert een dergelijk net?

Een **Hopfield-net** is een associatief geheugen gebaseerd op polaire TLU's met drempelwaarde 0. Daarbij zijn geen expliciete invoer- of uitvoerneuronen gebruikt, maar zal een invoerpatroon ingesteld worden als uitvoer van de aanwezige neuronen (het aantal bits van het invoerpatroon moet dus overeen komen met het aantal neuronen in het netwerk). Hierna worden neuronen willekeurig aangeduid (waarbij ze hun uitvoerwaarde herberekenen), totdat het netwerk stabiel is (meerdere stabiele toestanden zijn mogelijk). [1] Bij het herberekenen van een neuron, worden de inkomende verbindingen geëvalueerd: het gewicht ervan wordt vermenigvuldigd met de waarde van het neuron. Deze inkomende verbindingen kunnen wegens de symmetrie van het netwerk in een later stadium echter als uitgaande verbinding functioneren.

De **energiefunctie** beschrijft de energie van het netwerk, wat men ook kan beschrijven als de maat voor hoe vreemd de toestand van het net is. De formule is als volgt:

$$E = -\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}u_i u_j$$

Om te bespreken hoe de energiefunctie zich gedraagt bij het omkippen van neuron k, splitsen we de sommatie naargelang de afhankelijk van het neuron:

$$E = -\frac{1}{2}\sum_{i\neq k}^{n}\sum_{j\neq i}^{n} w_{ij}u_i u_j - u_k\sum_{i\neq k} w_{ik}u_i$$

Als we nu de energiefunctie op moment na en voor het omkippen vergelijken, kunnen we concluderen dat de energie nooit zal afnemen (hoogstens gelijk blijven, bij een overgang van -1 naar 1). Bijgevolg kan een systeem nooit teruggaan naar een toestand waarin het al geweest is, en zal het uiteindelijk in een stabiele toestand terechtkomen.

[1] Doordat we willekeurig een neuron aanduiden, is het netwerk niet-deterministisch!

Om een dergelijk net te doen **leren**, passen we regel van Hebb toe, waarbij de om de probabiliteit van een bepaald patroon te verhogen, de gewichten ervan aanpassen zodat de energie daalt:

$$w_{ij} \rightarrow w_{ij} + u_i u_j$$

Concreet zullen we bij het aanleren van een patroon, in geval van een + de gewichten van de verbindingen naar dat neuron met 1 verhogen, en anders diezelfde gewichten met 1 verlagen.

P109 - Beschrijf hoe je met een associatief netwerk kan groeperen zonder supervisie.

Aangezien Hopfield-netten gegeven een bepaalde invoer evolueren naar een stabiele toestand, kunnen we die stabiele toestand (waarvan er steeds minder zijn dan het aantal invoerpatronen) interpreteren als zijnde een classificatie van de gegevens. Om dit de realiseren zonder supervisie kunnen we volgend algoritme gebruiken:

1. Zet alle gewichten van het net op 0.
2. Train het geheugen met de gehele verzameling van punten
3. Fixeer de gewichten van het net, en verwerk nu opnieuw alle punten om vervolgens de stabiele toestand van het netwerk na het verwerken van een punt te interpreteren als de classificatie.

Aangezien er hier geen "fouten" kunnen optreden, is het nutteloos om de leerverzameling herhaald te presenteren: de gewichten zouden er gewoon bij verdubbelen.

P110 - Bespreek de omzetting van woord naar betekenis met behulp van associatieve geheugens.

Hiervoor kunnen we de **patroon-vervolledigingsmogelijkheden** van een Hopfield-net gebruiken. Daartoe moet eerst een gepaste methode gevonden worden om woorden om te zetten naar een bitpatroon, waarbij belangrijk is dat vergelijkbare woorden vergelijkbare bitpatronen oplevert. Eenmaal dat gedaan is, kunnen we woorden als invoer gebruiken bij een Hopfield-net, en de uitvoer beschouwen als de veralgemening daarvan, waarvan we de betekenis reeds kennen.

Hoofdstuk 12

Kennisrepresentatie in neurale netten

P112 - Hoe slaat men complexe kennis binnen neuraal netwerk op?

Eerst en vooral definiëren we hoe de gegevens gestructureerd zijn. Binnen een neuraal netwerk zoals onze hersenen, hebben we enkel beschikking tot knopen. Een knoop (of een ensemble daarvan) stelt een bepaald object voor, en kan geactiveerd of gedeactiveerd zijn. Hierbij merken we op dat de hersenen het activeren of deactiveren van knopen synchroniseert door gebruik te maken van synchronisatiegolven, waardoor we het denken kunnen definiëren als een opeenvolging van activatiepatronen.

Om **types en hun eigenschappen** voor te stellen baseren we ons op de theorie van het semantische net. Knopen kunnen daarbij direct in het neurale netwerk voorgesteld worden, geëtiketteerde verbindingen vormen echter een probleem. Hoe moeten we bijvoorbeeld uitdrukkingen realiseren? Het grootste probleem hierbij zijn uitdrukkingen die bestaan uit meerdere types knopen eventueel gecombineerd met eigenschappen (vb. "de zwarte kat achtervolgt een kleine bruine hond"):

- Combinaties voorstellen door unieke type-eigenschappen paren: deze methode zorgt wel voor een eenduidige classificatie, maar kent capaciteitsproblemen (voor elke mogelijke combinatie moet een knoop voorzien worden). Ook leert deze methode niet goed, en zitten we nog altijd in de problemen in geval van meerdere tokens met hetzelfde type.

- Toevoegen van identificatieknopen (die gebonden worden aan een token): hiermee kunnen makkelijk verschillende combinaties maken (vb "zwarte.a kat.a achtervolgen kleine.b bruine.b hond.b"), maar los van het feit dat we daardoor gelimiteerd zijn door het vaste aantal identificatieknopen, is dit niet in overeenstemming met de klassieke leerregels. Ook hebben we een probleem van alignatie.

Bij gebrek aan goede combinatiemogelijkheid, schrijven we voor dat een activatiepatroon zich moet beperken tot een enkel type.

Nu we types en hun eigenschappen kunnen voorstellen, moeten we nog definiëren hoe we **predikaten** in het systeem opslaan. Aangezien we per activatiepatroon slechts 1 type mogen gebruiken, moeten we een manier vinden om over de activatiepatronen heen de rollen van tokens te definiëren. Hiertoe bestaan verschillende denkpistes:

- Volgorde van voorkomen: hierbij wordt de rol toegekend gebaseerd op de volgorde van het voorkomen van het token. Het probleem is hierbij dat activatiepatronen cyclisch voorkomen, en het dus moeilijk is om een strikte orde te definiëren.
- Toevoegen van niet-semantische knopen: dergelijke knopen drukken de rol uit van het token. Ze zijn niet semantisch, want ze komen niet letterlijk voor binnen het semantische net (maar wel als de directionele component van een verbinding).

Deze laatste methode laat ons toe om eenvoudige predikaten uit te drukken. Het laat ons ook toe om verschillende gedachten over dezelfde types te verwerken: de eigenschappen dienen hiertoe als differentiatie (bij gebrek eraan worden de algemene eigenschappen het ene en het andere gebruikt). Voor complexere uitdrukkingen is het echter te beperkt, bij mensen wordt dit gerealiseerd in het werkgeheugen.

Tenslotte moeten we ook nog een wijziging doorvoeren aan de standaard manier van **leren**: aangezien ons neuraal net sequentiële denkpistes moet kunnen memoriseren, zal onze leerregel niet meet perfect symmetrisch mogen zijn, in tegenstelling tot de regel van Hebb die we tot nu toe hebben gebruikt. Daarom wijzigen we de regel als volgt:

$$w_{ij}(t+ \rightarrow w_{ij}(t) + u_i(t)u_j(t+))$$

Hierdoor is de regel afhankelijk van de invoer op het moment $t+1$, wat een asymmetrie in functie van de tijd introduceert. Daardoor zullen we in staat zijn sequentiële gegevens te onthouden.

Tenslotte introduceren we nog **samengestelde types**, die ons zullen toelaten een is een relatie te introduceren binnen ons netwerk. Daartoe maken we nieuwe types aan, die bestaan uit verschillende andere (niet noodzakelijk enkelvoudige) types.

P122 - Wat is de *suspension of disbelief*, en hoe komt ze voor binnen ons neuraal net?

Bij de suspension of disbelief denken en weten we twee contradictorische zaken. Hierbij moeten we eerst beide begrippen definiëren: **denken** is het actief zijn van bepaalde patronen binnen ons neuraal netwerk, terwijl **weten** het aanwezig zijn van bepaalde informatie. Beide fenomenen interageren: als we iets weten zullen we sneller geneigd zijn er over te denken, en anderzijds leidt het herhaald denken over een zaak tot het weten ervan.

Om de suspension of disbelief te kunnen verklaren, moeten we verduidelijken dat neuronen op verschillende manieren van elkaar kunnen verschillen. Zo is er de duurzaamheid, die bepaald hoe lang een bepaalde verbinding bewaard blijft. Dit resulteert in het onderscheid tussen het **langetermijn-geheugen** en het **kortetermijn-geheugen**. Een

ander verschil tussen die twee geheugens is echter de sterkte van de signalen. Hierdoor kunnen twee patronen die elkaar normaal tegenspreken, toch tegelijkertijd voorkomen: het patroon binnen het kortetermijn-geheugen zal dan echter het andere patroon overstemmen. Daardoor kunnen we tijdelijk denken aan bepaalde informatie, terwijl een tegenspraak in ons permanent net opgeslagen is. Logisch kunnen we dit ons voorstellen door een tijdelijk net binnen het permanente net.

P123 - Leg het verschil uit tussen *abstractie* en *onderscheid*. Hoe manifesteert zich dit?

Abstractie laat ons toe om van een bepaald type over te gaan naar een meer algemene uitspraak, door de gemeenschappelijke verbindingen tussen twee uitspraken (die handelen over verschillende types) te versterken. Dit fenomeen laat ons toe om vragen binnen het net te beantwoorden: door te denken aan een patroon gevormd door elementen van het antwoord die in de vraag aanwezig zijn, kunnen we de overige verbindingen van de vraag vervolledigen door stelselmatige activatie van de gemeenschappelijke verbindingen.

Onderscheid daarentegen komt voor wanneer we denken aan twee identieke types deelnemen aan verschillende gedachten. Hierbij worden de gemeenschappelijke verbindingen niet zozeer versterkt bij gebrek aan gemeenschappelijke gedachte, maar zullen de specifieke verbindingen zich het sterkst manifesteren. Hierdoor krijgen we tijdelijke types waarbij de specifieke eigenschappen zorgen voor differentiatie (bij gebrek hieraan worden de generieke eigenschappen het ene en het andere toegepast).

P125 - Hoe filteren BogoFilter en SpamAssasin spam?

BogoFilter is gebaseerd op puur Bayesiaans rekenen, waarbij men deze statistiek zal gebruiken om te bepalen of een bericht als spam moet geclassificeerd worden of niet. De eerste stap hierbij is het programma initialiseren met een set aan spam en niet-spam berichten (een leerverzameling), waarbij op voorhand geweten is hoe ze de klasseren. Hiermee zal BogoFilter een lijst met woorden samenstellen, met bijhorende informatie hoeveel spam en hoeveel niet-spam berichten dit woord bevatten. Gebaseerd hierop zal men voor een onbekend bericht kunnen bepalen wat de kans is dat het spam is.

Om dit principe te verbeteren, kunnen we de applicatie duidelijk maken welke berichten hij **verkeerd geklasseerd** heeft. Met deze informatie kan BogoFilter de factoren die bepalen of de aanwezigheid van een bepaald woord een hoge probabiliteit van spam veroorzaakt, verlagen. Dit is eigenlijk te vergelijken met een perceptron-netwerk, waarbij de regel van Hebb gewichten zal aanpassen indien er verkeerde classificatie optrad.

De applicatie **SpamAssasin** daarentegen maakt gebruik van een regelsysteem met onzekerheidsfactoren, waardoor conflictresolutie eenvoudiger wordt (door gebruik te maken van eenvoudige combinatiefunctie die, vergelijkbaar met een neuraal net, de som neemt en ze afweegt tegen een drempelwaarde).