Afstudeerproject Bachelor AI

Door :

Nicolaas Heyning
en
Wouter Suren

P2P Data Mining

Project begeleider:
Maarten van Someren

Nicolaas Heyning
1e Van der Helststraat 57-II
1073 AD, Amsterdam
nheyning@gmail.com

Wouter Suren
Simon Stevinstraat 3 huis
1097 BW, Amsterdam
wouter_wouter@yahoo.com

Universiteit van Amsterdam , 1 juli 2005

Samenvatting

In dit project hebben we onderzoek gedaan naar de werking en het resultaat van P2P data mining. Het P2P netwerk leert op basis van een naive Bayes classifier. De agents in het P2P netwerk worden per ronde een train sample aangeboden. Nadat ze hiervan geleerd hebben communiceert iedere agent met een andere agent. We hebben het P2P netwerk getest op basis van verschillende factoren. Zo hebben we geëxperimenteerd met het aantal agents in het netwerk en het aantal train cases. Onze bevindingen hieruit zijn dat het netwerk goed presteert. Het netwerk berijkt steeds sneller een hoge presentatie naarmate het aantal agents of het aantal train caes verhoogd wordt. Het netwerk hebben we vervolgens uitgebreid zodat het kan clusteren. Ook de prestatie van het clusterende netwerk hebben we onder verschillende factoren getest. Onze bevindingen zijn dat het clusteren goed werkt. Wat opvalt is dat na lange tijd clusteren het gevaar ontstaat dat de clusters naar elkaar toe groeien, dat wil zeggen, dat ze steeds meer elkaars data gaan representeren. We hebben een methode bedacht waardoor dit proces vertraagd wordt, en dus de clusters langer hun eigen data blijven representeren.
P2P data minen blijkt een snelle techniek te zijn die hoge prestaties levert. Het netwerk kan ook uitstekend uitgerust worden met clustering technieken.

Inhoudsopgave

1 Inleiding

Het doel van deze opdracht is het ontwerpen, bouwen en testen van een prototype van een systeem dat bestaat uit een aantal lerende agents, die elk een data mining taak uitvoeren door steeds nieuwe data te verzamelen en hiervan te leren. Daarnaast kunnen de agents peer-to-peer informatie uitwisselen. In dit geval gaat het bijvoorbeeld om welke features relevant zijn.
We hebben een aantal onderzoeksvragen die we willen behandelen. Als eerste zullen we een P2P naive Bayes classifier maken welke op een eenvoudige dataset zal leren. De dataset bestaat uit verschillende klassen en het is de bedoeling dat een agent na het trainen een nieuwe, onbekende case in de correcte klasse zal plaatsen. We zijn dan geïnteresseerd in hoe snel en of de agents zullen convergeren naar een gewenst resultaat.
Verder willen we onderzoeken of het mogelijk is om agents op verschillende soorten datasets uit hetzelfde domein te trainen. Hiervoor moeten de agents kunnen clusteren zodat er voor iedere dataset een groep ontstaat die goed op deze dataset kan classificeren. Een vorm van clustering is het meer overnemen van de waarde van een andere agent naarmate de agents meer op elkaar lijken. Dit alles willen we ondersteunen met een theoretisch model dat we willen vormen op basis van het P2P leren van een proportie. In hoofdstuk 2 zal de theoretische context toegelicht worden. In hoofdstuk 3 behandelt het theoretisch model van het P2P leren van een proportie. In Hoofdstuk 4 hebben we het over onze implementatie van de naive Bayes P2P classifier en de resultaten ervan. In hoofdstuk 5 behandelen we het P2P data minen met cluster technieken en behandelen we de verkegen resultaten. Hoofdstuk 6 is de conclusie van ons onderzoek.

Nicolaas heeft hoofdstuk 2,3 en 4 geschreven. Wouter heeft hoofdstuk 5 geschreven. Samen hebben we de rest van het verslag geschreven. Nicolaas heeft de P2P naive Bayes clasifier geïmplementeerd, Wouter heeft het clusteren en het herschrijven van de datasets (de strippers) geïmplementeerd.

2 Theoretische context

Data mining is het geautomatiseerd vinden van correlaties in hele grote datasets. Een data mining techniek heeft geen semantisch begrip van de data, maar doet het meer op basis van impliciete factoren die verschillende klassen kunnen hebben. Het wordt vaak gebruikt om verbanden te vinden die voorheen niet bekend waren, zoals het herkennen van fraudeurs uit creditcard gegevens. Data mining is gegroeid als een directe consequentie van de aanwezigheid van steeds grotere hoeveelheden digitale data door de jaren heen. Dit is begonnen in de jaren ’60 en heeft met de ontwikkeling van relationele databases met gestructureerde zoektalen (SQL) in de jaren ’80 een grote ontwikkeling doorgemaakt. Met de geboorte van het internet in de jaren ’90 is de groei van de data exponentieel toegenomen. Hierdoor zijn het aantal toepassingen van data mining erg breed (van zoekmachine tot het herkennen van potentiële klanten) en is er grote vraag naar efficiënte en nauwkeurige data mining technieken.
Er zijn momenteel veel verschillende data mining technieken, deze variëren voornamelijk in de manier waar op ze de analyse representeren. Dit kan bijvoorbeeld gebeuren in de vorm van een beslisboom, een neuraal netwerk of als conditionele kansen. De techniek die gebruikt wordt is vaak afhankelijk van wat voor soort data geanalyseerd moet worden.
In ons onderzoek zullen wij gebruik maken van een naive Bayes classifier. Dit omdat het een makkelijk te implementeren techniek is en omdat, zoals we later zullen laten zien, verschillende modellen van een naive Bayes classifier goed te combineren zijn.

Een naive Bayes classifier werkt op basis van conditionele kansen. Gegeven een test case wil men bereken wat de kansen zijn op de verschillende klassen. De testcase wordt dan geclassificeerd als de klasse met de hoogste kans. Omdat de kans op een klasse, gegeven een case, onbekend is zetten we deze kansen om volgens de regel van Bayes zodat we factoren krijgen die we wel kunnen berekenen.

$$P(klasse|case) = \mathrm{P}(case|klasse) \cdot \frac{\mathrm{P}(klasse)}{\mathrm{P}(case)} \tag{2.1}$$

P(klasse) is de a a-priori kans dat een klasse voorkomt.
P(case|klasse) is de conditionele waarschijnlijkheid dat een testcase voorkomt, gegeven een bepaalde klasse.
P(case) is de kans op een bepaalde testcase. Deze is onafhankelijk van alle klassen en is dus constant. Daarom kan dit bij het berekenen van een kans op een klasse weggelaten worden.

De kans op een klasse wordt dan als volgt berekend:

$$P(klasse|case) = \mathrm{P}(case|klasse) \cdot \mathrm{P}(klasse) \tag{2.2}$$

Om te kunnen data minen moet een naive Bayes classifier eerst getraind worden. Dit bestaat uit het opbouwen van relatieve frequentie tabellen van de a a-priori en de conditionele kansen van een grote training set. Met deze kansen kunnen we dan berekenen tot welke klasse een nieuwe case het meest waarschijnlijk behoort. Het is belangrijk dat de training data representatief is voor de toekomstige data die geleverd wordt. Vaak wordt de geleverde training data opgedeeld in driekwart training data en een kwart test data, om te testen of het model de juiste resultaten geeft.
Het opbouwen van de relatieve frequentie tabellen kan gezien worden als het werk dat een agent doet. Tot op het heden wordt er bij het data minen alleen nog maar gebruik gemaakt van slechts

een agent. Het kan echter zijn voordelen hebben om dit werk te verdelen over meerdere agents. Ten eerste komt het tegenwoordig vaak voor dat datasets zo groot zijn (zoals het internet) dat het voor een agent te veel werk is om dit allemaal alleen te modelleren. Dit kan betekenen dat het te veel tijd kost of dat de agent de computationele resources mist om op alle data te trainen. Door meerdere agents tegelijk te laten werken kan dit opgeloste worden, omdat de tijd en de resources verdeeld worden. Ten tweede kan het voorkomen dat niet alle data samen gevoegd is, maar bijvoorbeeld om geografische- of privacy redenen uit losse stukken bestaat. Ziekenhuizen bezitten bijvoorbeeld vaak een database waar alle gegevens van de patiënten in opgeslagen zijn. Uit deze gegevens zouden nuttige conclusies getrokken kunnen worden, zoals het verband tussen bepaalde symptomen en een ziekte. Hoe meer data hiervoor gebruikt wordt hoe beter het model kan worden. Het is echter voor ziekenhuizen niet mogelijk om patiëntengegevens uit te wisselen, omdat hiermee de privacy van de patiënten wordt geschonden. Data mining met meerdere agents zou hier dan goed van pas komen, door losse agents op de data te laten minen en die dan de geëxtraheerde informatie, die niet privacy gevoelig is, te laten uitwisselen zodat er wel over het geheel geleerd wordt.

Er zijn verschillende manieren om meerdere agents samen te laten werken. Je kunt meerdere agents gebruik laten maken van dezelfde geheugen bank waar ze dan iedere keer hun vergaarde kennis in opslaan zodat alle andere agents deze kunnen overnemen, de zogenaamde blackboard methode. Dit heeft echter een paar beperkingen, ten eerste brengt dit een groot risico met zich mee, omdat als deze centrale geheugen bank uitvalt of storing heeft dan heeft dit effect op alle agents. Ten tweede zal deze aanpak langzaam zijn, omdat al de te trainen data op een plek verwerkt zal moeten worden, hetgeen computationeel erg zwaar en inefficiënt is.

Een betere aanpak is het P2P samenwerken tussen agents. Dit houdt in dat er geen centraal punt is waarlangs alle informatie gaat, maar de agents onderling informatie uitwisselen. ‘Gossip-based’ protocollen zijn een makkelijke manier van communicatie waarbij iedere agent na iedere getrainde case met één willekeurige andere agent zijn informatie uitwisselt. Zo wordt de informatie op een veel efficiëntere manier verspreid, en zal het netwerk ook minder gevoelig zijn voor storingen of het uitvallen van een element.
Twee agents die parallel leren en hun resultaten uitwisselen moeten hetzelfde resultaat hebben als één agent die alleen over alle data leert omdat ze alle informatie iedere keer met elkaar uitwisselen. Als er meer dan 2 agents zijn dan blijven dezelfde resultaten alleen behouden als alle agents iedere keer alles met elkaar communiceren. Dit is echter niet altijd haalbaar, omdat de communicatie kosten erg hoog zijn. Als de agents ‘gossip-based’ communiceren dan zullen ze wel naar elkaar toe convergeren maar een slechter resultaat hebben omdat de trainingsinformatie niet meer uniform verdeeld is en er dus wat verloren gaat. Echter, hoe meer trainingsdata de agents krijgen, hoe beter de resultaten zullen worden.

Clusteren is het proces van het herkennen en verdelen van subklasses van gelijke soort in een dataset. Deze subklasses heten clusters. Alle data items uit een cluster behoren tot dezelfde groep. De bedoeling is dat de mate van gelijkheid binnen een cluster maximaal is, terwijl de verschillende clusters onderling een minimale gelijkheid hebben. De data in clusters representeren heeft het noodzakelijke gevolg dat er verlies van de fijne details optreedt, immers er treedt een vorm van generalisatie op. Het zoeken naar clusters is een vorm van unsupervised learning. Dit betekent dat het zoeken naar patronen gebeurt zonder het sturen of beoordelen van buitenaf (bijvoorbeeld de mens). Clusteren is een techniek die uitstekend toegepast kan worden op data mining applicaties zoals Web analyse, wetenschappelijk data onderzoek, tekst mining, marketing en medische diagnoses.

3 Theoretisch model van het P2P leren van een proportie

De theorie voor het P2P leren van een proportie is gebaseerd op een model dat Niko Vlassis1 heeft onderzocht. In zijn model was de bedoeling dat agents op een P2P manier het gemiddelde van een reeks getallen konden berekenen. Ze gaan hier uit van het newcast model waarbij iedere subverzameling van de dataset een agent is. Dit betekent dat ieder getal in de getallen reeks een agent is. Per iteratie zoekt iedere agent dan een willekeurig andere agent, de waarde van deze twee agents worden dan geüpdate met het gemiddelde van de twee agents. Uit het onderzoek bleek dat een dataset van 105 getallen al na 41 iteraties convergeerde naar het juiste gemiddelde.
Dit model kan ook geïmplementeerd worden door het aantal getallen dat een agent kan hebben te vergroten. Dit kan op twee manieren.
Zo kan iedere agent bij initialisatie al de benodigde getallen hebben. De agent berekent hiervan een gemiddelde en communiceert dit per cyclus met een willekeurige andere agent. Ook dit zal convergeren, omdat het gemiddelde van gemiddelden gelijk is aan het gemiddelde van de hele reeks getallen, en de variantie van de reeks afneemt naarmate hij uit meer gemiddelden bestaat.
Op de andere manier krijgt iedere agent per cyclus een nieuw getal, berekent vervolgens een nieuw lokaal gemiddelde op basis van dit nieuwe getal en een oud gemiddelde2. De agent communiceert dit gemiddelde met een willekeurige andere agent. Dit convergeert ook tot een correct gemiddelde, omdat dit weer gemiddelden van gemiddelden zijn en het aantal waarden (agents) niet verandert, terwijl de variantie van de waarden wel afneemt.

Wat wij willen weten is of dit ook mogelijk is voor het leren van een proportie. Men moet het experiment als volgt voor zich stellen. Stel, we hebben N agents en we hebben een hele grote bak met rode en zwarte knikkers. Iedere agent krijgt per cyclus een knikker uit de bak te zien, welke daarna weer terug gelegd wordt. Het is dan de bedoeling dat de agents een schatting maken van de proportie van de bak knikkers. Het is interessant om proporties te leren, omdat een naive Bayes classifier ook gebaseerd is op proporties, en het voor het P2P leren dus belangrijk is dat de agents hun ervaring kunnen combineren. Stel we hebben één agent. De geschatte proportie rode knikkers is dan gebaseerd op het aantal voorkomens rode knikkers gedeeld door het totaal aantal tegengekomen knikkers.
Als er meerdere agents zijn dan moeten die allemaal hun eigen proportie berekenen op basis van de geobserveerde knikkers, maar ook per iteratie moet iedere agent zijn proportie uitwisselen met een willekeurige andere agent. Het komt er op neer dat het gemiddelde gevonden wordt van de proporties. Het is de bedoeling dat de resultaten van alle agents naar elkaar convergeren, zodat het resultaat van iedere individuele agent gelijk wordt aan het resultaat van het leren over de totale dataset. Dit is wel te verwachten, omdat als men een serie proporties heeft, en twee van deze proporties worden vervangen door het gemiddelde van de twee, dan verandert het totale gemiddelde niet. Wat wel verandert is de variantie van deze reeks. Deze wordt exponentieel lager2. Stel namelijk een X en Y die onafhankelijke willekeurige variabelen zijn welke een waarde aannemen uit V, de verzameling proporties van alle agents, waarbij deze waarden allemaal dezelfde kans hebben.

[1] W. Kowalczyk, N. Vlassis . *Newscast EM*, ftp://ftp.science.uva.nl/pub/computer-systems/aut-sys/reports/Kowalczyk05nips.pdf
[2] W. Kowalczyk, M. Jelasity, A.E. Eiben, *Towards Data Mining in Large and Fully Distributed Peer-to-Peer Overlay Networks*, http://www.dcs.napier.ac.uk/~benp/dream/dreampaper19.pdf

Dan

$$\mathrm{E}\left[\frac{X+Y}{2}\right] = \mathrm{E}[X] = \mathrm{E}[Y] = \mathrm{E}[V] \tag{3.1}$$

en

$$\mathrm{VAR}\left(\frac{X+Y}{2}\right) = \frac{\mathrm{VAR}(X)}{4} + \frac{\mathrm{VAR}(Y)}{4} = \frac{\mathrm{VAR}(Y)}{2} \tag{3.2}$$

Waarbij *E*[ ] de verwachte waarde (het gemiddelde) is en *VAR*( ) de variantie van een willekeurige variabele is.
Om de analyse wat te versimpelen gaan we er van uit dat per cyclus maar een paar agents kan communiceren. Dit zorgt ervoor dat het proces vertraagd zal worden, maar uiteindelijk leidt dit wel tot hetzelfde resultaat. Als er N agents zijn, dan zal het N keer zo lang duren voordat alle agents evenveel met elkaar hebben gecommuniceerd.
Stel een verzameling W die gevormd wordt door 2 willekeurige elementen uit de verzameling V te vervangen door hun gemiddelde. De verwachte waarde van de variantie van W na een iteratie wordt dan bepaald door:

$$\mathrm{E}[\mathrm{VAR}(W)] = \left(1 - \frac{1}{N}\right) \cdot \mathrm{VAR}(V) \tag{3.2}$$

Dit betekent dat per iteratie *k* de afname van de variantie van de P2P berekende dataset afneemt met de factor $\left(1-\frac{1}{N}\right)^{K}$. Als het aantal iteraties dan groot genoeg is dan zal de variantie naar nul gaan en is het juiste gemiddelde berekend. Het bewijs is als volgt;
V is de verzameling van de geschatte proporties van de agents, μ is het gemiddelde van V, N is het aantal agents, W is de verzameling van de geschatte proporties van de agents nadat 2 agents hebben gecommuniceerd. We nemen een χ en een γ voor twee willekeurig geselecteerde elementen uit V. Dan hebben we:

$$N \cdot (\mathrm{VAR}(V) - \mathrm{VAR}(W)) = (\chi - \mu)^2 + (\gamma - \mu)^2 - 2 \cdot \left(\frac{\chi + \gamma}{2} - \mu\right)^2 = \tag{3.3}$$
$$= \chi^2 + \gamma^2 - 0{,}5 \cdot (\chi + \gamma)^2$$

Als we vervolgens de χ en de γ vervangen voor de willekeurige variabelen X en Y, die de selectie van χ en γ modelleren, dan krijgen we:

$$\begin{aligned} & N \cdot \mathrm{E}[\mathrm{VAR}(V) - \mathrm{VAR}(W)] = \\ & 0.5 \cdot \mathrm{E}[(X-Y)^2] = 0.5 \cdot (\mathrm{E}[X^2] + \mathrm{E}[Y^2] - 2 \cdot \mathrm{E}[X] \cdot \mathrm{E}[Y]) = \\ & = \mathrm{E}[V^2] - \mu^2 = \mathrm{VAR}(V) \end{aligned} \tag{3.4}$$

En daarom is $\mathrm{E}[\mathrm{VAR}(W)] = \left(1 - \frac{1}{N}\right) \cdot \mathrm{VAR}(V)$

Deze functie kan dan als volgt geïnterpreteerd worden; De gemiddelde afwijking van de juiste proportie door agents is afhankelijk:

- Van het aantal agents. Hoe meer agents, hoe langer het duurt voordat ze convergeren, omdat ze maar met een andere agent tegelijk kunnen communiceren. Dit is alleen waar als er per iteratie maar 1 train case aan de agents gegeven wordt en er maar 1 agent actief kan leren.
  Als per iteratie iedere agent een train case krijgt, dan kunnen de agents parallel leren en zal de variantie veel sneller afnemen.
- De variantie van de training set waarop V is gebaseerd (het verschil tussen de klassen). Bij een meer gevarieerde training set zullen de agents meer verschillend worden, waardoor ze langzamer zullen convergeren.
- Het aantal iteraties. Hoe meer iteraties, hoe kleiner de gemiddelde afwijking van iedere agent wordt en hoe meer ze op de daadwerkelijke proportie lijken.

4 De P2P naive Bayes classifier

In dit deel van ons onderzoek willen we onderzoeken of het mogelijk is om een P2P naive Bayes classifier te maken, en wat de resultaten hiervan zijn.
Zoals in het vorige hoofdstuk is gezegd is het mogelijk statistische berekeningen te verdelen over verschillende agents zonder dat dit het uiteindelijke resultaat zal beïnvloeden. Als er maar genoeg iteraties plaatsvinden waarbij de agents allemaal een traincase krijgen dan zullen ze uiteindelijk naar een correct model van de data convergeren. En als de trainingsdata dan representatief is voor de testdata zal dit tot juiste classificaties leiden.
In hoofdstuk 2 werd al uitgelegd dat een naive Bayes classifier werkt door het berekenen van de kansen op een bepaalde classificatie gegeven een te classificeren case. De case zal dan geclassificeerd worden als de klasse met de grootste kans.

$$\text{Classificatie} = \arg\max_{\text{klasse}} P(\text{klasse}|\text{case}) \qquad (4.1)$$

Voordat we konden beginnen met het implementeren van de classifier zijn we op zoek gegaan naar een dataset waarop we konden trainen en testen. We hebben hiervoor gebruik gemaakt van de resources van de UCI machine learning repository[3]. Dit is een databank van de Universiteit van California met grote datasets toegankelijk zijn. We hebben gekozen voor een dataset bestaande uit de handgeschreven getallen 0 tot en met 9. Een case ziet eruit als in figuur 4.1.

```
00000000000000000111100000000000
00000000000000001111110000000000
00000000000000001111110000000000
00000000000000001111110000000000
00000000000000011111110000000000
00000000000000011111110000000000
00000000000000111111100000000000
00000000000001111111000000000000
00000000000001111111000000000000
00000000000011111110000000000000
00000000000011111110000001000000
00000000001111111000001111000000
00000000011111110000011111100000
00000001111111100000111111100000
00000001111111000000111111100000
00000011111110000000111111100000
00000011111111111111111111100000
00000111111111111111111111000000
00001111111111111111111111000000
00001111111111111111111110000000
00000111111111111111111100000000
00000011111111111111111100000000
00000001111110000111111100000000
00000000000000000111111100000000
00000000000000001111111000000000
00000000000000001111111000000000
00000000000000011111110000000000
00000000000000011111110000000000
00000000000000111111000000000000
```

4

Figuur 4.1

[3] http://www.ics.uci.edu/~mlearn/MLRepository.html

De features van de cases zijn de indexen van de bits, iedere feature kan dus de waarde 0 of 1 hebben. Dit brengt twee voordelen met zich mee. Ten eerste is het aantal features altijd gelijk, namelijk 1024. Ten tweede is een naive Bayes classifier makkelijker te implementeren met binaire features. Dit komt omdat dan alleen maar gekeken hoeft te worden of een feature positief (1) is of niet en er hoeft geen rekening gehouden te worden met verschillende waarden van elke feature, aangezien de kans op een 0 gelijk is aan 1 min de kans op een 1. In totaal zijn er 1934 cases welke we hebben opgedeeld in een training set van 1521 en een testset van 413. De dataset bevat geen missende data. De voorkomens van de klassen zijn gelijk verdeeld en hebben allemaal een waarschijnlijkheid tussen 0 en 1. We hebben een stripper gemaakt de trainingsdata inleest en deze aan ons programma, case voor case doorgeeft.

We zijn eerst begonnen met het maken van een standaard naive Bayes classifier, die in zijn eentje de hele dataset doorwerkt. Zoals in de theoretische context is uitgelegd wordt de conditionele kans van de classifier berekend op basis door het product te nemen van de a-priori kans en de conditionele kans van een case gegeven een klasse (zie functie (2.2))

De a-priori kans en de conditionele kans worden tijdens het trainen opgebouwd door het berekenen van de relatieve frequenties van de trainingsdata.

P(klasse) = aantal voorkomens van de klasse/totaal aantal voorkomens van alle klassen.

Een case wordt gekenmerkt door zijn features, dus de kans op een case wordt opgebouwd door de kansen op de feature waarden:

$P(case|klasse) = \prod P(feature_i|klasse)$ voor alle feature waarden i in case

P(feature|klasse) = het aantal positieve voorkomens van een feature uit een case/aantal voorkomens van de klasse.

Tijdens het trainen worden alle cases langsgegaan en de voorkomens van de klassen geteld en voor iedere klasse worden de voorkomens van alle features bijgehouden. Met deze twee modellen kan de tester dan te werk gaan. Om een test case te classificeren wordt de kans op alle waardes van de features voor iedere mogelijke klasse berekend, gegeven de trainingsdata. Dit betekent dat voor de 10 mogelijke klassen $P(case|klasse)$ wordt berekend door $\prod P(feature_i|klasse)$ van de klasse te berekenen. Voor alle tien de klassen worden deze waarden vermenigvuldigd met P(klasse). Dit geeft de kans dat een classificatie correct is gegeven de case. De classificatie met de hoogste kans wordt dan gegeven als de correcte classificatie. Een agent kan dan nu een score krijgen door te kijken welk percentage van de testdata correct is geclassificeerd.
Onze implementatie had met deze data een score van 88.33%, wat een goede score is maar gegeven de structuur en de uniformiteit van de data niet heel erg onverwachts is.

Vervolgens zijn we begonnen aan een P2P implementatie van onze naive Bayes classifier. Zoals gezegd willen we gebruik maken van een ‘gossip-based’ protocol voor de communicatie tussen de agents, wat inhoud dat we per iteratie iedere agent één willekeurige andere partner laten zoeken waarmee hij zijn gegevens over de net geleerde klasse uitwisselt. Nu zal duidelijk zijn waarom wij voor een naive Bayes implementatie van een classifier hebben gekozen in plaats van een neuraal netwerk of een beslisboom. Omdat bij naive Bayes classifier er maar weinig gegevens uitgewisseld hoeven te worden en deze makkelijk te combineren zijn. De enige informatie die nodig is voor een naive Bayes classifier, en die dus uitgewisseld moet worden, is per klasse een frequentie tabel voor de features in die klasse en een frequentie tabel van de voorkomens van die klasse.

Deze kunnen gecombineerd worden door een gemiddelde te nemen tussen de waarden van beide agents. Omdat de feature tabel én de klasse voorkomens worden gemiddeld zal het resultaat van de nieuwe P(case|klasse) gelijk zijn aan de P(case|klasse) van één agent die over op beide trainsets samen heeft geleerd. Namelijk:

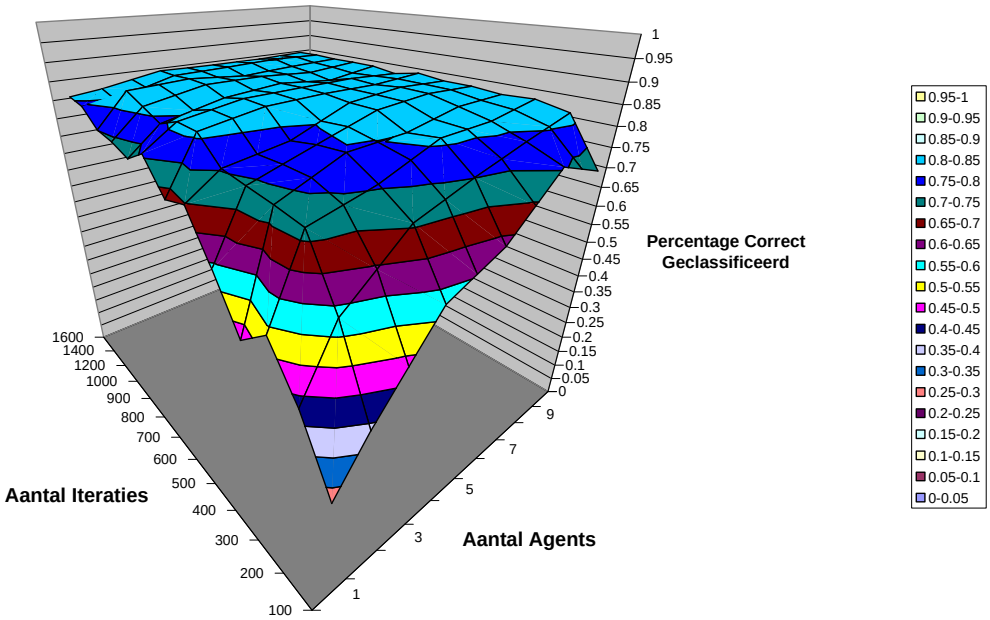
$$\frac{(Case1+Case2)/2}{(Klasse1+Klasse2)/2} = \frac{(Case1+Case2)}{(Klasse1+Klasse2)}$$

Waarbij Case1 en 2 de feature tabellen zijn.

P(klasse) blijft ook nog een correcte waarde houden, omdat het totaal aantal voorkomens van de klassen onveranderd blijft. Dit komt omdat de som van het aantal voorkomens van een bepaalde klasse voor twee agents gelijk is aan twee keer de som van het gemiddelde van deze voorkomens. De som van de a a-priori kansen van de verschillende klassen blijft gelijk aan 1.

Het is nu goed te zien dat het parallel laten minen van meerdere agents leidt tot betere resultaten, aangezien N agents N keer zoveel informatie verwerken. Er is echter een beperking. Als er meer dan twee agents zijn dan zullen de resultaten suboptimaal zijn. Omdat de agents per iteratie met slechts een andere agent communiceren, zal de geleerde informatie over alle cases niet volledig verspreid worden. De communicatieparen die willekeurig gekozen worden garanderen dat de agents bij veel iteraties evenveel met elkaar informatie uitwisselen. Toch zal er altijd wat informatie verloren gaan, hetgeen een nadeel kan zijn van het P2P data minen.
Omdat we maar een beperkte hoeveelheid data hebben, hebben we voor veel agents maar een klein aantal traincases. Om dit te verkomen, en ervoor te zorgen dat alle agents toch verschillende trainingsdata hebben, laten we de stripper nu een willekeurige case uit de training geven. Dit kan dan oneindig vaak gebeuren. Hierdoor wordt de trainingsdata echter minder uniek en zullen de test resultaten slechter zijn. Met één agent en 1600 trainingsiteraties behalen we nu een resultaat van 80 %.

We hebben een aantal tests gedraaid waarbij we het aantal agents en het aantal training iteraties laten variëren. De locatie van de data is te vinden in Appendix A. Figuur 4.2 geeft een plot van de data weer.



Figuur 4.2

Zoals in figuur 4.2 is te zien heeft bij een vast aantal traincases het aantal agents een positieve invloed op de score. Hoe meer agents in het netwerk aanwezig zijn, hoe beter het netwerk scoort. Zoals verwacht heeft het aantal iteraties dat het netwerk leert ook een positieve invloed, omdat de agents dan meer training data krijgen. Ook bij een zeer hoog aantal iteraties presteert het netwerk met meerdere agents beter dan een enkele agent. Het effect van meer agents, en van meer training iteraties, neemt echter wel af en convergeert uiteindelijk naar een maximale prestatie ongeveer 85%. Dit komt omdat de relatieve frequenties van een agent na meer iteraties steeds minder beïnvloedt zullen worden door nog een extra positief voorbeeld.

5 Toepassing van clusteren bij P2P data mining

Clusteren wordt vooral gebruikt om verborgen patronen te herkennen in een dataset, dat wil zeggen, om meer betekenis aan de data te geven. Clusteren kan ook noodzakelijk zijn als agents hetzelfde domein bestuderen, maar waarbij de data van het domein opgedeeld is in verschillende soorten groepen. Denk maar aan data van bepaalde ziektes, die wordt bijgehouden door ziekenhuizen in verschillende landen. Het is goed mogelijk dat, ondanks dat de ziekenhuizen data over dezelfde ziektes bijhouden, de data structureel verschilt. Dit kan zijn omdat de patiënten van ziekenhuis A op een totaal andere manier leven dan de patiënten uit ziekenhuis B. In dit geval is het niet ondenkbaar dat patiënten uit het ene ziekenhuis met andere oorzaken dezelfde symptomen (lees: ziektes) krijgen als de patiënten uit het andere ziekenhuis. Als de data van beide ziekenhuizen tot een dataset wordt samengevoegd en men laat er een groep agents op leren, dan bestaat het gevaar dat de onderlinge verschillen tussen de symptoomgroepen verdwijnen. De agents zullen na het trainen de data van beide ziekenhuizen representeren, en dus onnauwkeurigere uitspraken kunnen doen over een patiënt, omdat er niet meer rekening gehouden kan worden met uit welk gebied de patiënt afkomstig is.
Als de agents cluster technieken toepassen, zullen groepen agents data afkomstig uit een bepaald gebied gaan overnemen, zodat er clusters ontstaan. De clusters zullen specifieker een bepaald gebied beschrijven dan een groep agents die de totale data beschrijft. Hierdoor zullen agents die een bepaald gebied representeren de patiënten uit die regio beter kunnen classificeren (diagnosticeren).

Ons netwerk van agents is ook getest op een vorm van clusteren. Het clusteren vindt plaats tijdens de train fase. Bij elke iteratie wordt iedere agent een train sample gegeven. De agents zullen aan de hand van de train sample hun bestaande kennis aanpassen. Daarna wordt er met een andere agent gecommuniceerd. Dit gebeurt volgens een cluster techniek.

Voordat we clusteren konden toepassen moesten we de dataset verdelen in verschillende groepen die door de agents gerepresenteerd konden worden. We kozen ervoor om de cijfers in de dataset te verschuiven. Zoals gezegd wordt een cijfer in de dataset gerepresenteerd als een item met 1024 attributen met voor elk attribuut de waarde nul of een. Door alle attributen van een item een x aantal op te verschuiven, en door attributen op het eind (die eruit vallen) weer aan het begin te zetten, onstaat er een verschoven getal. De verschoven dataset representeert dus dezelfde cijfers, maar aan de hand van andere features. De helft van de agents zullen data uit de niet verschoven dataset krijgen, andere helft krijgt data uit de verschoven data set.
Nu de agents leren uit verschillende domeinen kunnen ze wel met elkaar blijven communiceren, maar ze communiceren in feite over verschillende conclusies, wat het gevolg is van verschillende statistische trainkennis.

Het cluster gedrag van het netwerk wordt door middel van twee parameters bepaald en kan dus ook middels deze twee parameters veranderd worden. Deze twee parameters zijn:

- Similarity
- Influence

Similarity

Het cluster effect wordt toegepast op het moment dat de twee agents tijdens het communiceren van elkaar gaan leren. Op het moment dat een agent A met agent B in contact staat en de waardes van de tabel bekend zijn, wordt eerst berekend in hoeverre beide agents het met elkaar eens zijn. Deze waarde heet de similarity. De similarity wordt uitgedrukt in een proportie variërend tussen 0 en 1.

De similarity wordt als volgt berekend. Er zijn twee frequentie tabellen bekend, tabel A (van agent A) en tabel B (van agent B). Een tabel bestaat uit 1024 attributen, die elk een frequentie hebben. (zie tabel 5.1a en 5.1b). Van elk attribuut wordt de verhouding tussen dit attribuut uit tabel A en tabel B berekend. Om te voorkomen dat deze verhoudingen boven 1 komen wordt altijd de kleinste waarde gedeeld door de grootste waarde. (zie tabel 5.1c) Tot slot wordt het gemiddelde van alle 1024 verhoudingen genomen. Dit is de gemiddelde verhouding tussen de attributen van de tabel A en de attributen van de tabel B en is tevens de similarity. (zie Tabel 5.1)

De minimale similarity is de waarde 0. Dit is een uiterste dat zelden voorkomt. In dit geval is er voor elk van de 1024 attribuut in een van de twee tabellen de waarde 0 gevonden. De maximale similarity is 1. Dan zijn alle attributen aan elkaar gelijk. Een similarity van 0,2545 wil zeggen dat tabel A en tabel B voor 25% aan elkaar gelijk zijn.

Attribuut nummer	…	**565**	**566**	**567**	**568**	**569**	**570**	**571**	**572**	…
Frequentie	…	13	10	40	34	10	11	5	3	…

Tabel 5.1a. *Een deel van tabel A*

Attribuut nummer	…	**565**	**566**	**567**	**568**	**569**	**570**	**571**	**572**	…
Frequentie		10	10	43	30	8	12	2	0	

Tabel 5.1b. *Een deel van tabel B*

Attribuut nummer	…	**565**	**566**	**567**	**568**	**569**	**570**	**571**	**572**	…
Verhouding		10/13	10/10	40/43	30/34	8/10	11/12	2/5	0/3	
Similarity = 10/13 + 10/10 + 40/43 + 30/34 + 8/10 + 11/12 + 2/5 + 0/3) / 8 = 0.7123										

Tabel 5.1c. *De verhoudingen en bijbehorende similarity. Voor dit deel uit de tabellen geldt dus dat ze voor 71% gelijk aan elkaar zijn.*

Influence

Naast de similarity is ook een andere parameter verantwoordelijk voor het clusteren, de influence. Bij het niet clusteren wordt altijd de helft van de andere agent met de helft van de eigen waardes opgeteld. De influence bepaalt hoe zwaar een agent de meetresultaten van de andere agent laat meetellen. Deze waarde stelt ‘de mate van vertrouwen’ van de agent voor. De influence wordt van tevoren handmatig bepaald en mag variëren tussen 0 tot 1. Een influence van 0 heeft tot gevolg dat de agent voor 0% naar de andere agent mag luisteren. Bij een influence van 1 luistert de agent maximaal voor 100% naar de andere agent. Hoeveel er dan geluisterd wordt bepaald de vooraf berekende similarity.

Het product van de similarity en de influence vormen de proportie die de agent van de andere agent in zijn meetwaarden meeneemt. Omdat beide variabelen variëren tussen nul en een is deze

waarde te beschouwen als de mate die een agent de andere agent beïnvloeden. Is dit product hoog, bijvoorbeeld 0.78, dan wordt 78% van de andere agent en 22% van zichzelf de nieuwe waarde voor de agent. Als het product boven de 0,5 komt betekent dit dat de agent meer naar de andere agent gaat luisteren dan naar zichzelf. Om dit te voorkomen kunnen we de influence op 0,5 zetten, zodat het product nooit boven de 0,5 kan komen (immers, bij maximale similarity wordt het product 1 * 0,5 = 0,5). De influence garandeert dus dat er nooit meer dan deze proportie van de andere agent meegenomen kan worden. Zie tabel 5.2 voor een voorbeeld. Met een similarity van 0,7123 en een influence van 0,5 wordt er 35% van de andere agent meegenomen in de eigen meetwaardes.

- Similarity = 0.7123
- Influence = 0.5
- Proportie andere agent = 0.7123 * 0.5 = 0.3561
- Proportie van zichzelf = 1-0.3561 = 0.6439

Attribuut nummer	…	**565**	**566**	**567**	**568**	**569**	**570**	**571**	**572**	…
Proportie		8.3707	6.439	25.756	21.8926	6.439	7.0829	3.2195	1.9317	

Tabel 5.2a. *Proportie van Agent A. Iedere frequentie waarde wordt met 0,6439 vermenigvuldigd*

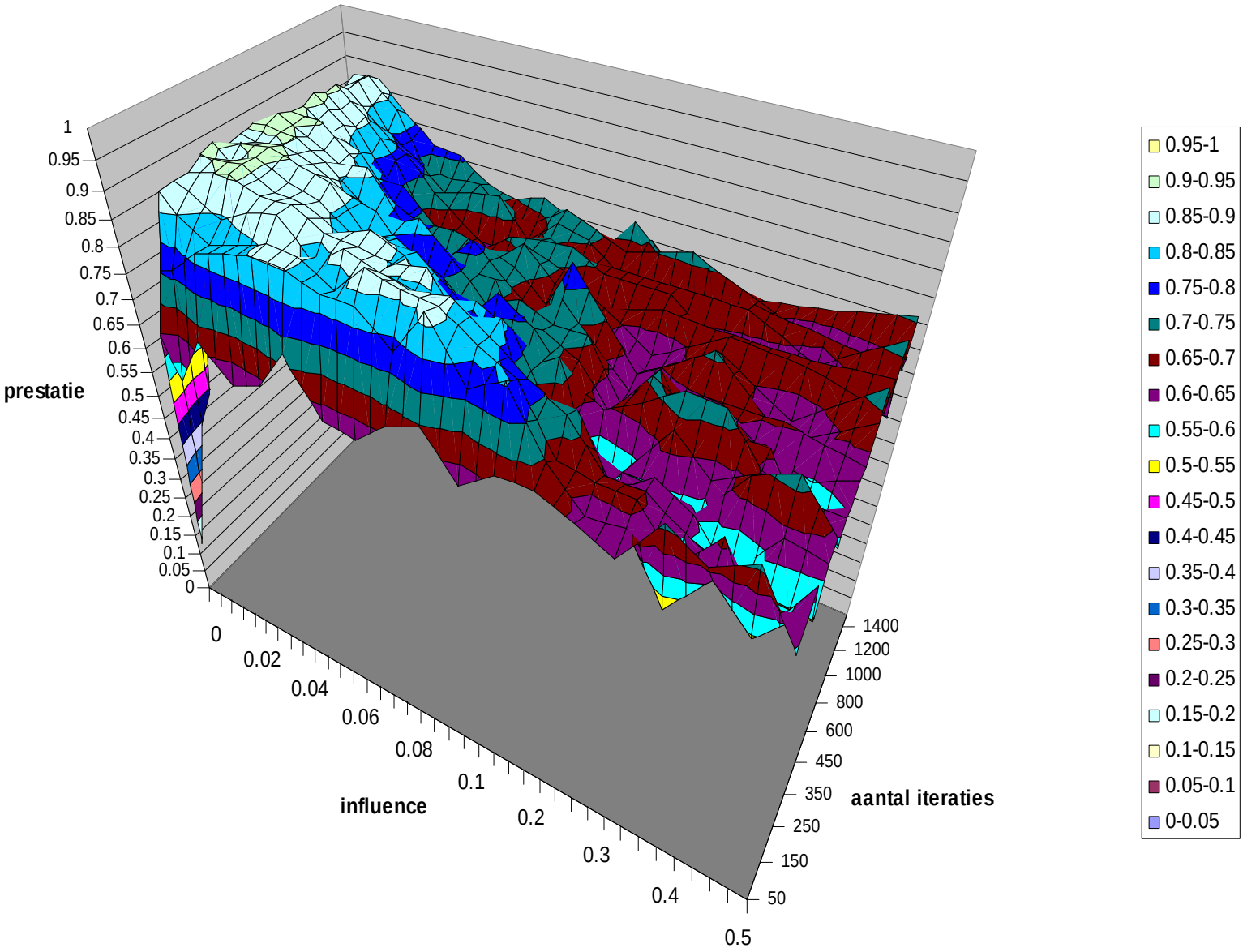
Attribuut nummer	…	**565**	**566**	**567**	**568**	**569**	**570**	**571**	**572**	…
Proportie		3.561	3.561	15.3123	10.683	2.8488	4.2732	0.7122	0	

Tabel 5.2b. *Proportie van Agent B. Iedere frequentie waarde wordt met 0,3561 vermenigvuldigd*

Attribuut nummer	…	**565**	**566**	**567**	**568**	**569**	**570**	**571**	**572**	…
frequentie		11.9317	10	41.0683	32.5756	9.2878	11.3561	3.9317	1.9317	

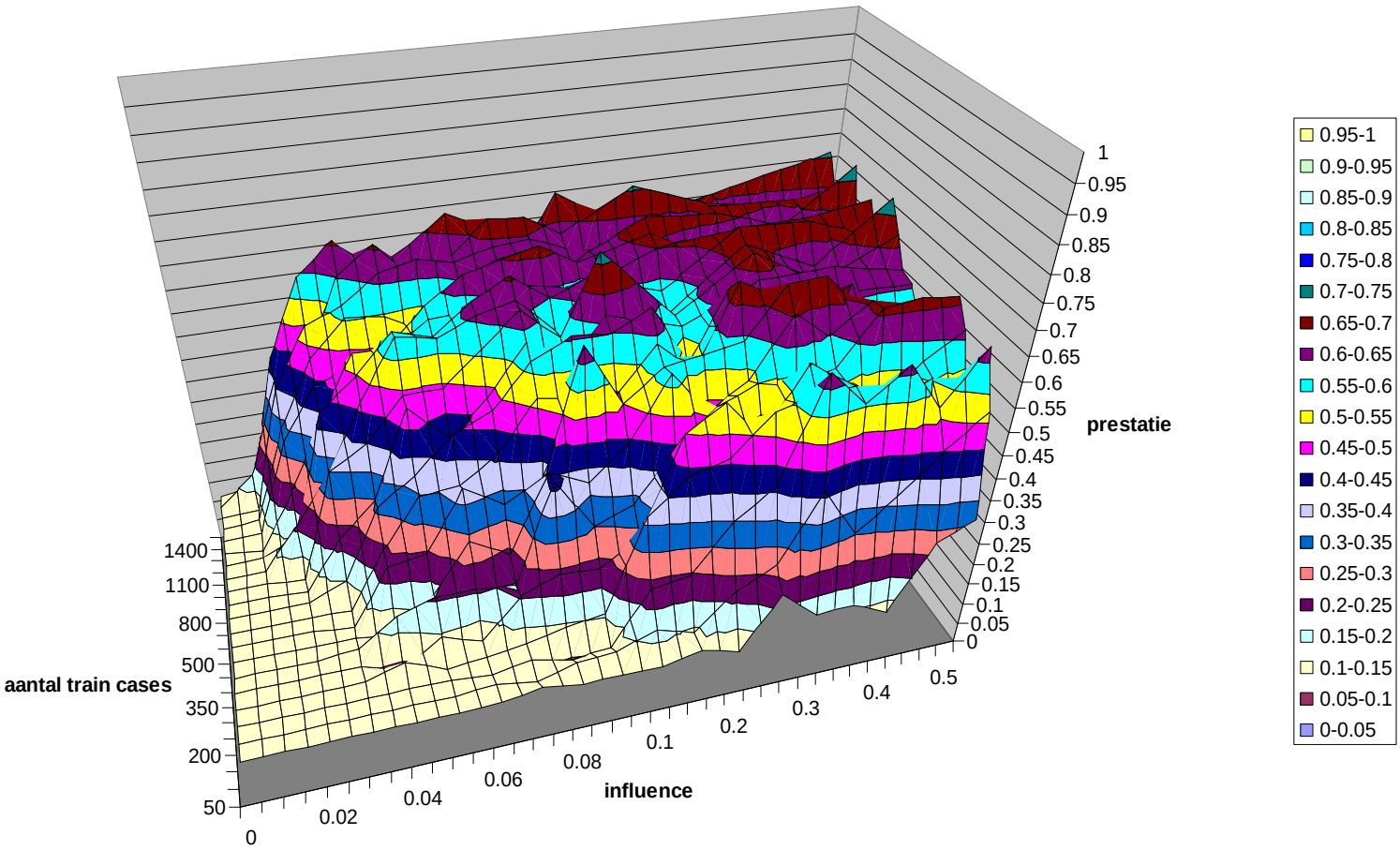
Tabel 5.2c. *Nieuwe frequenties voor de attributen voor agent A.*

We hebben een data set gecreëerd waarbij we de helft van alle cases met een offset van 10 hebben opgeschoven. De data werd hierdoor dusdanig gevarieerd dat als we er P2P op gingen data minen zonder clustering, zoals in hoofdstuk 4, dat we na 1500 iteraties een score van 60% kregen. Vervolgens zijn we op deze data set het cluster effect gaan testen door het aantal iteraties en de influence van de agents met clusteren te variëren. Het aantal agents hebben we ingesteld op 4, door het aantal agents laag te houden is de verbetering in de score langzamer zodat we de invloeden van de overige factoren beter kunnen zien.



Figuur 5.1 *Prestatie van de clusters getest op de data van de eigen cluster*

Het eerste wat ons opviel is dat de influence heel erg laag moet zijn om toch een goed resultaat te halen. Daarom hebben we bij het testen vooral gefocust op een influence tussen 0 en 0.1. In de plot hebben we dit ook zo weer gegeven. Het is belangrijk dat u rekening houdt met de ongelijke verdeling van de schaal als u deze plots leest! We hebben hiervoor gekozen omdat nu duidelijk te zien is hoe het kritieke gebied verloopt en we toch een goed beeld krijgen van hoe het totale beeld verloopt.
Zoals in figuur 5.1 te zien is zorgt een influence van rond de 0.008 voor een optimaal resultaat. Een dergelijk lage influence is nodig omdat de datasets behoorlijk van elkaar verschillen en de agents elkaar erg verkeerde data zouden kunnen geven. Bij veel iteraties zal deze informatie toch verspreid raken en beïnvloeden de agents elkaar. De agents die op de test set getraind zijn krijgen dus slechtere resultaten, wat te zien is in figuur 5.1. Opvallend is dat het resultaat naar de 60% toe gaat, wat gelijk is aan de score van het P2P data minen zonder clustering. Als de influence nul is zijn de resultaten veel slechter. Dit komt omdat bij nul influence de agents helemaal niet communiceren, waardoor ze veel langzamer leren met als gevolg een lagere score.

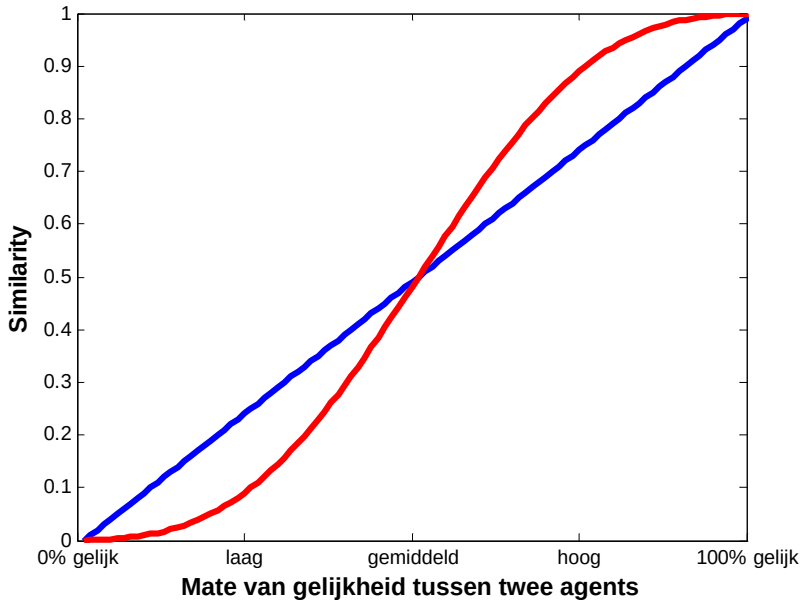


Figuur 5.2 *Prestatie van de clusters getest op de data van de andere cluster*

In figuur 5.2 is duidelijk het tegenover gestelde te zien. Ook hier moet weer rekening gehouden worden met de non-lineaire schaalverdeling. Omdat deze agents op een andere dataset zijn getraind dan dat ze zijn getest hebben ze in het begin een hele slechte score. Zodra de influence begint door te werken begint het resultaat van de deze agents toch te verbeteren omdat ze leren van de kennis van de agents die wel op de juiste set leren. De omslag is snel, omdat de proportie waarmee agents van elkaar leren ook afhankelijk is van hun similarity. Dus hoe meer ze op elkaar lijken, hoe meer ze van elkaar leren.

Verbetering van het clusteren: De sinusoïde

Een probleem is dat bij een grotere influence of bij het aanbieden van een groter aantal train cases, de clusters langzaam naar elkaar toe gaan groeien. Er zijn verschillende manieren om dit tegen te gaan. Een manier om dit effect te remmen is om te zorgen dat als twee communicerende agents minder op elkaar lijken, dat er relatief minder naar elkaar wordt geluisterd. Evenzo wil je dat agents die het met elkaar eens zijn meer van elkaar leren. Op dit moment gedraagt de mate van similarity zich lineair met de mate van gelijkheid tussen de agents (zie figuur 5.3).



Figuur 5.3. *De manier waarop de mate van gelijkheid tussen twee agents gemapt wordt naar een similarity waarde. Er kan een lineair verband zijn (blauw) of een sinus verband zijn (rood).*

De oplossing is het herberekenen van de verkregen similarity met een functie die de bovengenoemde methode simuleert (zie figuur 5.3). De volgende functie doet dit.

$$y = A \cdot \sin(2\pi \cdot x) + x \tag{5.1}$$

Dit is een sinus functie, die alle punten tussen nul en een kan herberekenen naar een waarde die voldoet aan de bovengenoemde voorwaarde. A is de amplitude. Deze moet zo bepaald worden dat het bereik $y \in [0,1]$ blijft. Een goede waarde voor de amplitude is - 0.15. Het is als volgt te bewijzen.

$$\begin{aligned} y(x = 0) &= 0 \\ y(x = 1) &= 1 \end{aligned} \tag{5.2}$$

Nu moet bewezen worden dat de functie altijd stijgt tussen x = 0 en x = 1.

$$y' = -0.15 \cdot 2\pi \cdot \cos(2\pi \cdot x) + 1 = 0 \tag{5.3}$$

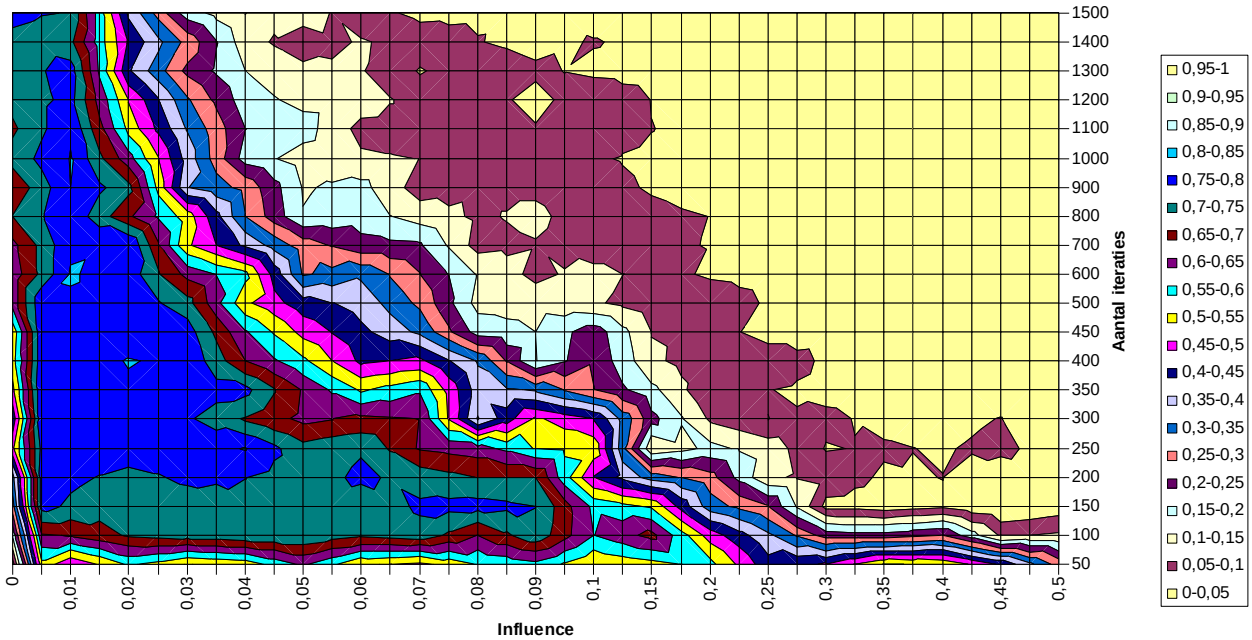
$$\cos(2\pi \cdot x) = \frac{-1}{-0.15 \cdot 2\pi} > 1 \qquad (5.4)$$

Er bestaat geen x dit hieraan voldoet. cos (2 π * x) kan nooit een waarde groter dan 1 opleveren. Er zijn dus geen nulpunten. Vul nu een willekeurige waarde voor x en vul deze in bij y'.

$$y'(0,5) = -0.15 \cdot 2\pi \cdot \cos(2\pi \cdot 0,5) + 1 = 1.9424 \qquad (5.5)$$

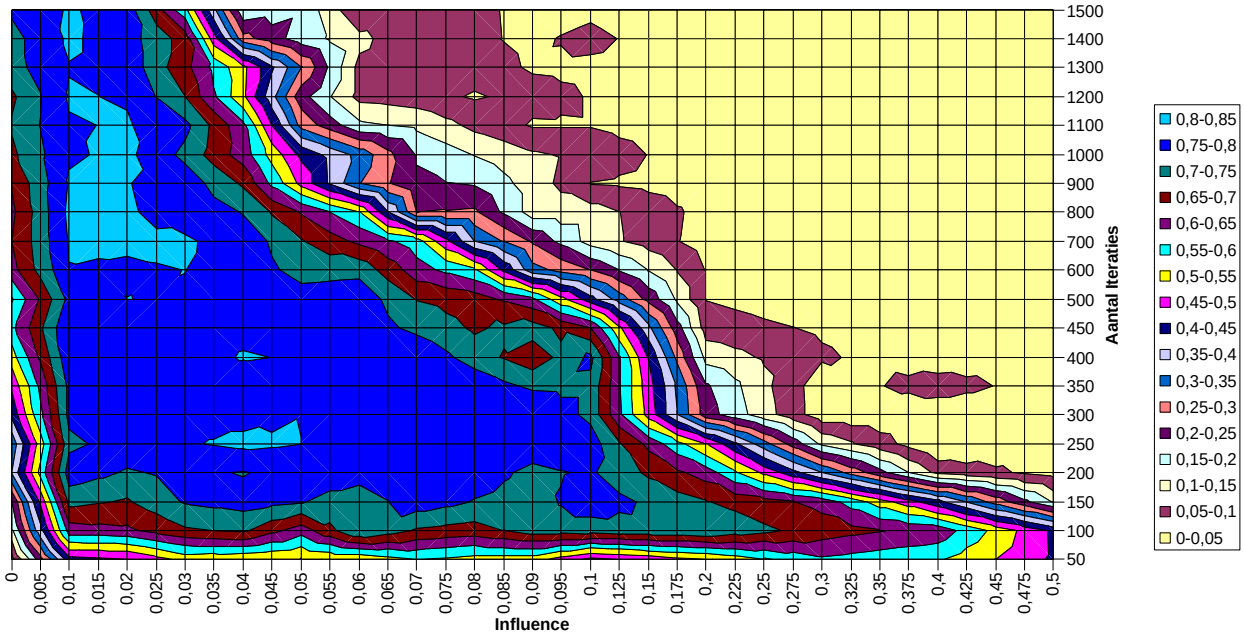
De afgeleide (5.3) is altijd positief, (5.1) stijgt voor alle x, dus ook voor het domein x ∈ [0,1]

Deze nieuwe similarity functie hebben we geïmplementeerd en op dezelfde manier getest als de oude functie. De resultaten van de beide clusters hebben we van elkaar afgetrokken en in een grafiek weergegeven.



Figuur 5.4 *Resultaten van de oude similarity functie. Bovenaanzicht van het verschil in performance tussen de clusters die op hun getrainde data zijn getest en de clusters die niet op hun getrainde data zijn getest.*

In Figuur 5.4 is te zien dat als de influence laag is dat dan het verschil tussen de scores van de clusters groot is. Dit betekent dat de cluster die op dezelfde set wordt getest als dat hij wordt getraind veel beter scoort dan de andere cluster. Daarnaast is er een verband te zien tussen de influence en het aantal iteraties. Als er veel iteraties plaats vinden dan kan er alleen nog goed gescoord worden bij een lage influence. Als het aantal iteraties afneemt dan kan de influence toenemen om toch nog een goed resultaat te houden.
Opvallend is dat het verschil uit twee uitersten bestaat en dat de overgang hiertussen er snel verloopt. Dit komt omdat de clusters elkaar meer beïnvloeden naarmate ze meer op elkaar gaan lijken. Als het clusteren te lang door gaat, door te veel iteraties (en een te hoge influence), dan gaan de clusters steeds sneller elkaar meer beïnvloeden. Als de clusters eenmaal gelijk zijn, dan blijft het verschil constant rond de nul.



Figuur 5.5 *Resultaten van de nieuwe similarity functie. Bovenaanzicht van het verschil in performance tussen de clusters die op hun getrainde data zijn getest en de clusters die niet op hun getrainde data zijn getest.*

In figuur 5.5 is wederom het verschil in de performance tussen de twee clusters geplot. Dit keer is er bij het data minen echter gebruik gemaakt van de verbeterde similarity functie. Het eerste dat opvalt, is dat het cluster dat getraind en getest wordt op dezelfde set veel langer optimaal blijft presteren. Zowel bij een hogere influence als bij meer iteraties blijft het verschil tussen beide cluster groot. Dit komt omdat de nieuwe similarity functie bij grote ongelijkheid tussen de cluster een lager getal geeft, waardoor de beïnvloeding meer onderdrukt wordt. Verder is er te zien dat de overgang tussen de uitersten een stuk kleiner is geworden. Dit komt omdat de nieuwe similarity functie in dit gebied snel stijg van invloed onderdrukken naar invloed versterken (zie figuur 5.3). Als de clusters eenmaal weer gelijk zijn dan blijft het verschil weer constant rond de 0.

6 Conclusie

We hebben in dit project onderzoek gedaan naar de werking en het resultaat van P2P data mining. Data mining is de techniek waarbij men grote hoeveelheden data analyseert om verbanden te vinden die men op het eerste gezicht niet ziet. Een data mining techniek heeft geen semantisch begrip van de data, maar doet het meer op basis van impliciete factoren die verschillende klassen kunnen hebben. Tot op het heden wordt er bij data mining gebruik gemaakt van 1 agent die op alle data getraind wordt om zo een goed model te krijgen wat gebruikt kan voor het analyseren van nieuwe data. Dit kan echter nadelen hebben omdat de data bijvoorbeeld te groot is om door een agent te verwerken (het internet). Of omdat de data om geografische- of privacy redenen niet samen te voegen is. Door het gebruik van meerdere agents kunnen deze problemen voorkomen worden. Iedere agent leert dan zelfstandig op zijn eigen data en geeft deze kennis door aan de andere agents zodat deze ook van elkaar leren. Dit kan door gebruik te maken van een centrale databank waar alle kennis in opgeslagen wordt en die door elke agent toegankelijk is. Dit brengt echter hoge communicatie kosten met zich mee en kan een risico vormen als dit centrale punt uitvalt. Een andere mogelijkheid is het P2P communiceren tussen de agents. Dit houdt in dat de agents direct van elkaar leren, zonder aanwezigheid van een centrale kennis bank.
De theorie achter het P2P data minen is dat als men een serie waardes heeft, in dit geval de leer resultaten van een agent, dat men deze waardes kan middelen zonder dat het gemiddelde over al deze waardes zal veranderen. De variantie over al deze waardes wordt wel minder wat betekent dat alle agents naar hetzelfde gemiddelde convergeren. Dit gemiddelde is gelijk aan de waarde die één agent alleen ook zal leren.
Er zijn verschillende data technieken mogelijk zoals een beslisboom, een neuraal netwerk of een naive Bayes classifier. We hebben voor de laatste gekozen omdat deze gemakkelijk te implementeren is. Verder zijn de resultaten makkelijk te combineren omdat de naive Bayes classifier gebruikt maakt van conditionele kansen die goed te middelen zijn.
De data die we gebruikt hebben bestaat uit een reeks handgeschreven getallen tussen de nul en de negen, gerepresenteerd in 1024 bits (features).
Wij hebben in dit project de invloed van een aantal factoren op het P2P data mining onderzocht. In eerste instantie hebben wij gekeken in hoeverre het minen met meer dan 1 agent het resultaat verbetert. Vervolgens hebben we gekeken naar de mogelijkheid om agents te laten clusteren in het geval van verschillende data groepen uit hetzelfde domein.
Nadat we een implementatie hebben gemaakt van een naive Bayes classifier die zelfstandig werkt hebben we een P2P implementatie gemaakt. Deze geeft iedere agent per iteratie een willekeurige training case uit de data. Op basis hiervan houdt iedere agent een relatieve frequentie tabel bij van het aantal voorkomens van de features en van het aantal voorkomens van de klassen. Na iedere iteratie zoeken de agents een willekeurige andere agent en wisselen hun informatie over de net geleerde klasse uit. Dit doen ze door hun eigen waardes te vervangen voor het gemiddelde van de beide waardes.
Dit hebben we getest met een verschillend aantal iteraties en met een verschillend aantal agents. De agents worden getest op een test set, en hun score is gebaseerd op het aantal correcte classificaties van de onbekende cases. Het resultaat van meer iteraties is dat de score van iedere agent hoger wordt. Dit is logisch, omdat iedere agent een beter model krijgt van de te classificeren klassen. Als het aantal agents wordt verhoogd, dan wordt de score per agent ook hoger. Dit komt omdat de agents de kennis van elkaar overnemen zodat ze per iteratie meer trainingsdata krijgen, en zo een beter model kunnen vormen van de te classificeren klassen. Als er meer dan twee agents zijn dan is het resultaat echter wel suboptimaal omdat er maar met 1 agent per iteratie gecommuniceerd kan worden en er voor de overige agents informatie verloren gaat.

Het is mogelijk dat de agents leren van data die bestaat uit verschillende subdomeinen. Als de agents van deze dataset leren door te middelen zullen de resultaten niet goed zijn. Om voor de verschillende agents een onderscheid te maken in de subdomeinen kan men gebruik maken van cluster technieken zodat er groepen agents ontstaan die op de subdomeinen passen.
We hebben het clusteren geïmplementeerd door agents die veel op elkaar lijken makkelijker met elkaar te laten communiceren. Agents die weinig op elkaar lijken nemen juist weinig informatie van elkaar over. Het resultaat hiervan is dat de agents in verschillende subdomeinen naar elkaar toe groeien en clusters vormen. De mate van communicatie wordt bepaald door twee factoren; de similarity, dit is de mate waarin de agents op elkaar lijken. De andere factor is de influence, dit is de mate waarin de agents elkaar maximaal kunnen beïnvloeden. Deze twee factoren worden met elkaar vermenigvuldigd en vormen de proportie waarin een agent de informatie van een andere agent overneemt
Om dit te testen hebben we in de helft van de data de features van de cases 10 bits opgeschoven. Hierdoor ontstond er een nieuwe set die genoeg verschilde van de originele, maar toch uit hetzelfde domein komt. We hebben de test uitgevoerd met een vast aantal agents, verschillende influence en een verschillend aantal iteraties. Uit het resultaat bleek dat de influence heel laag moet zijn om ervoor te zorgen dat agents uit verschillende clusters elkaar niet gaan beïnvloeden. Bij een hoog aantal iteraties werkt deze invloed echter altijd door en zullen de agents naar elkaar convergeren. Om het convergerende proces te vertragen hebben we de similarity functie aangepast zodat deze niet meer lineair is. We hebben hem vervangen voor een sinusoïde die ervoor zorgt dat de communicatie tussen agents die sterk verschillen harder wordt onderdrukt. De communicatie tussen agents de op elkaar lijken wordt door de sinusoïde juist versterkt. Het effect hiervan hebben we getest en het blijkt dat de performance van de clusters nu tot meer iteraties en een hogere influence goed blijft.

We hebben een aantal aspecten van het P2P data minen onderzocht en getest. De techniek biedt zeker veel perspectieven, zowel in snelheid als in prestatie. Er zijn echter nog veel factoren die onderzocht moeten worden voordat men optimaal gebruik kan maken van de P2P technieken. Ten eerste kan men nog het P2P data minen vergelijken met het multi-agent data minen waarbij men gebruik maakt van centrale kennisbank. Omdat er dan geen informatie verloren gaat tijdens het leren en de kennis goed verspreid wordt kan dit misschien tot betere resultaten leiden wat de nadelen opheft. Ook moet er nog getest worden met andere, minder uniforme informatie waardoor de robuustheid van het systeem beter getest wordt. De code zou ook nog efficiënter gemaakt kunnen worden zodat er met nog veel meer agents, grotere datasets en meer subdomeinen getest kan worden.
Een ander aspect is het unsupervised clusteren, waarbij van te voren nog niet bekend is in wat voor een subdomeinen de data verdeeld is. Dit kan geïmplementeerd worden door agents cases die sterk van hun huidige kennis afwijken af te wijzen en wellicht door te laten geven aan andere agents. Wat het clusteren ook nog zou kunnen verbeteren is het verlagen van de influence naar mate de agents meer training cases heeft gehad, omdat de agent dan zekerder is van zijn eigen kennis en zich minder door andere agents hoeft te laten beïnvloeden. Hierdoor voorkom je dat de ontstane clusters in een later stadium van het leren weer naar elkaar toe gaan groeien.

APPENDIX A – De data

De data van onze bevindingen en ook de data van onze grafieken zijn te groot in omvang om in het verslag te kunnen vermelden. De data is daarom online te vinden op de volgende adressen.

- Data van hoofdstuk 4:
  http://student.science.uva.nl/~nheijnin/data/P2P%20data%20zonder%20clusteren.xls
- Data van hoofdstuk 5:
  http://student.science.uva.nl/~nheijnin/data/P2P%20data%20met%20clusteren.xls