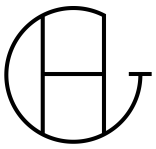


HoGent

Faculteit Bedrijf en Organisatie

Vergelijken voorspellingsalgoritmen met games

Matthias Seghers

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Nathalie Declercq

Instelling: ToThePoint

Academiejaar: 2016-2017

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Vergelijken voorspellingsalgoritmen met games

Matthias Seghers

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Nathalie Declercq

Instelling: ToThePoint

Academiejaar: 2016-2017

Tweede examenperiode

Samenvatting

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus.

Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Voorwoord

Inhoudsopgave

1 Inleiding 9

1.1 Stand van zaken 9

1.2 Probleemstelling en Onderzoeksvragen 10

1.3 Opzet van deze bachelorproef 11

2 Methodologie 13

2.1 Keuze algoritmen 13

2.1.1 Superviseerd vs Ongesuperviseerd leren 13

2.2 Gesuperviseerde classificatie algoritmen 14

2.2.1 Logistische regressie 14

2.2.2 Support Vector Machine 18

3 Uitwerking 19

3.1 Benodigdheden 19

3.1.1 Data 19

3.1.2 Frameworks 19

3.2 Fase 1 20

3.2.1 Data 20

3.2.2 Logistische regressie 21

4 Conclusie 25

Bibliografie 27

1. Inleiding

De inleiding moet de lezer alle nodige informatie verschaffen om het onderwerp te begrijpen zonder nog externe werken te moeten raadplegen (Pollefliet, 2011). Dit is een doorlopende tekst die gebaseerd is op al wat je over het onderwerp gelezen hebt (literatuuronderzoek).

Je verwijst bij elke bewering die je doet, vakterm die je introduceert, enz. naar je bronnen. In LaTeX kan dat met het commando `\textcite{}` of `\autocite{}`. Als argument van het commando geef je de "sleutel" van een "record" in een bibliografische databank in het BibTeX-formaat (een tekstbestand). Als je expliciet naar de auteur verwijst in de zin, gebruik je `\textcite{}`. Soms wil je de auteur niet expliciet vernoemen, dan gebruik je `\autocite{}`. Hieronder een voorbeeld van elk.

Knuth (1998) schreef een van de standaardwerken over sorteer- en zoekalgoritmen. Experten zijn het erover eens dat cloud computing een interessante opportuniteit vormen, zowel voor gebruikers als voor dienstverleners op vlak van informatietechnologie (Creeger, 2009).

1.1 Stand van zaken

De onderzoeken en technieken om de noden van gebruikers te voorspellen kent de laatste jaren een grote opmars. Google en Facebook zijn dan ook volop bezig met eigen onderzoekscentra en technologieën te ontwikkelen. FAIR is de afkorting voor Facebook Artificial Intelligence Research er zijn 3 labo's in de wereld die constant opzoek zijn naar nieuwe mogelijkheden binnen AI. RankBrain is een algoritme van Google die a.h.v. artificiële intelligentie de ranking op de zoekpagina bepaald. De persoonlijke advertenties die

Google toont zijn veelal gegenereerd met algoritmes. Maar ook gerelateerde producten of "wat jou ook kan interesseren=lijsten worden dikwijls door machine learning gegenereerd. Tensorflow („Tensorflow framework", g.d.) is een open source API ontwikkelt door Google die je uiteraard gratis kan gebruiken om zelf artificiële toepassingen te maken. De API is ontwikkeld in Python, deze taal is dan ook een van de veel gebruikte in de data science.

In deze bachelorproef zullen er voorspellingen gemaakt worden op een arcademachine. ToThePoint zal het eerste bedrijf zijn die AI in een arcademachine zal verwerken. Aangezien er nog geen andere gelijkaardige voorbeelden te vinden zijn zal ervan bij het begin onderzoek moeten geworden hoe we het best zouden aanpakken en verklaren. De spelletjes worden bestuurd door zes knoppen en een joystick. Aan de hand van de snelheid, aantal keer een knop of joystickbeweging is gedaan zullen we kunnen onderscheiden welk spel de gebruiker aan het spelen is. Er bestaan al veel verschillende soorten algoritmen maar deze zijn niet allemaal geschikt voor deze casus en daar zal dus aan gewerkt moeten worden.

Artificiële Intelligentie is hot en zo komen er allerlei beginnende frameworks naar boven die reeds geïmplementeerde algoritmen bevatten of het eenvoudiger maken om ermee te starten. Tensorflow is een populair framework ontwikkelt door Google gemaakt in Python. Dit wordt gebruikt in verschillende producten van Google zoals, Google's stemherkenning, Google vertaler, het zoeken op afbeeldingen, etc. TensorFlow werkt aan de hand van deep learning of neurale netwerken. De voorbeelden van Tensorflow gaan bijna uitsluitend over image recognition, het is mogelijk om andere applicaties ermee te maken. Dit is dan ook de reden dat Google dit framework open source heeft gemaakt zodat ze kunnen zien waar er nog verbeteringen kunnen aangebracht worden en wat er nog allemaal mogelijk is. Een ander interessant framework is het accord-framework („Accord framework", g.d.) die volledig ontwikkeld is in C#. Hierin zitten veel geïmplementeerde algoritmen in die dan kan gebruikt worden door derden om een eigen applicatie te ontwikkelen.

1.2 Probleemstelling en Onderzoeksvragen

Machine learning is momenteel aan het boomen. Dit wordt nu zeer veel toegepast voor online advertising met Google en Facebook als de leiders. Maar ook meer en meer bedrijven beginnen zich te verdiepen in artificiële intelligentie. ToThePoint is zich hierop ook aan het voorbereiden. Doormiddel van een funproject willen ze zoveel als mogelijk bijleren op allerlei gebieden binnen de informatica.

Men heeft een arcade machine gekocht die ze volledig gaan customizen met verschillende technologieën Op deze manier kunnen ze al hun kennis loslaten. Inclusief de kennis die ze zullen opdoen via deze bachelorproef. Hierdoor zullen ze dus iets bijleren over artificiële intelligentie en dit dan ook kunnen toepassen. Verder zal de machine geplaatst worden op allerlei jobbeurzen om gegevens van studenten op te slaan bijvoorbeeld. Het nut van de machine is niet alleen om bij te leren maar hij zal ook dienen als referentie naar klanten toe. Op deze manier kan ToThePoint aantonen tot wat ze instaat zijn.

In deze bachelorproef wordt er een vergelijkende studie gemaakt over twee verschillende

algoritmen. Het beste algoritme zal dan uiteindelijk geïmplementeerd worden in de arcade machine. Je kan hier heel ver in gaan. De voorspellingen kunnen bijvoorbeeld gemaakt worden door wat de meest gebruikte knop is of joystick beweging. Maar door de snelheid dat knoppen bediend worden en de frequentie van dezelfde knop kan bepaald worden welk spel er gespeeld wordt. Als het nog verder uitgewerkt wordt kan er naar toetsencombinaties gekeken worden maar dit is te vergaand voor deze bachelorproef. Er zal vooral gefocust worden op hoe een goed algoritme gekozen wordt. Het stappenplan die uitgewerkt zal worden kan dan ook toegepast worden op toekomstige projecten.

1.3 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.

In Hoofdstuk 4, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2. Methodologie

2.1 Keuze algoritmen

Om goede resultaten te behalen is het uiterst belangrijk dat de juiste algoritmen gebruikt worden. Zo zijn er bepaalde algoritmen die helemaal niet bruikbaar zouden kunnen zijn voor deze casus. Er zal hier verder uitgelegd worden hoe we een mogelijk algoritme kunnen vinden.

2.1.1 Superviseerd vs Ongesuperviseerd leren

Machine learning kan onderverdeelt worden in verschillende types van algoritmen. Deze zijn gesuperviseerd leren, ongesuperviseed leren en reinforcement leren. Dit laatste type is niet van toepassing voor deze casus. Met dit type wordt er geleerd op basis van positieve signalen (beloningen). Er wordt ook niet gebaseerd op een dataset en aangezien wij in bezit zijn van datasets is dit type overbodig om verder te onderzoeken.

Superviseerd leren

Dit type heeft als doel om een hypothese te bekomen die dan zal kunnen gebruikt worden om nieuwe ongekende input toe te wijzen aan een label die voorkwam in de eerdere trainingsdataset. De trainingsdataset bestaat uit verschillende parameters en een label. Onder dit type kan je algoritmen vinden die voor zowat alle cases gebruikt kunnen worden. Deze zijn echter wel nog opgedeeld in drie verschillende categorieën. Zo kan je een nieuwe waarde voorspellen op basis van vroegere resultaten, dit wordt regressie genoemd. Verder heb je classificatiealgoritmen hiermee kan een input toegewezen worden aan een

bepaalde klasse met een label. En als laatste bestaan er clusteringsalgoritmen hiermee kan je ook onderverdelingen maken in klassen maar deze hebben geen label. Clustering en classificatie lijken op elkaar maar met classificatie weet een onderzoeker ook precies wat de data voorstelt.

Ongesuperviseerd leren

Verschillend met gesuperviseerd leren beschikt een ongesuperviseerd algoritme over een ongelabelde dataset. Er zijn verschillende inputs maar die behoren niet tot een specifieke klasse. De meest gebruikte techniek is dus clustering. Als we dit bekijken voor deze casus is dit geen optimale manier. We kunnen wel bepaalde besturingsevents clusteren waardoor je tot een x aantal clusters kan komen maar we hebben geen idee of de ene cluster PacMan of Mortal Kombat is bijvoorbeeld.

misschien een voorbeeldje uitwerken?

2.2 Gesuperviseerde classificatie algoritmen

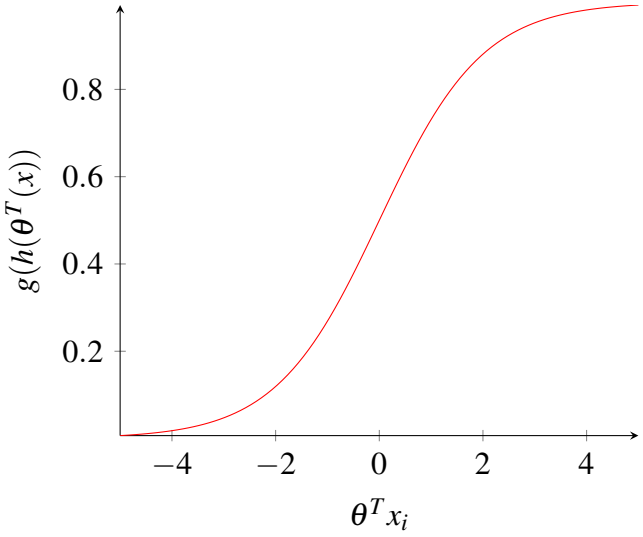
In deze casus beschikken we over een gelabelde dataset. Het doel is om met een gegeven input een concreet spel te krijgen als output. Gesuperviseerde classificatiealgoritmen zijn dus de meest geschikte voor deze in dit onderzoek. Doordat we een gelabelde dataset hebben en er moeten verdelingen gemaakt worden op basis van labels. Op deze manier kunnen we ongekende input plaatsen bij één bepaald spel.

2.2.1 Logistische regressie

Logistische regressie is het eerste algoritme die we zullen bespreken in deze bachelorproef. Ondanks de naam doet vermoeden, valt dit algoritme niet onder de categorie regressie zoals we in sectie 2.1.1 superviseerd-leren hebben gezien. Dit is wel degelijk een classificatiealgoritme die echter wel gebruik maakt van de logistische functie. Aan de hand van die hypothese kunnen er voorspellingen gemaakt worden. Gedurende het trainingsproces wordt de hypothese geoptimaliseerd. Er zullen meerdere spelletjes voorzien zijn op de arcademachine dus ook meerdere klassen. Omdat we van nul af aan starten zal ook de binaire logistische regressie gebruikt worden en vervolgens zal een van de twee mogelijke methoden om met meerdere klassen te werken gebruikt worden..

Binaire logistische regressie

Er moeten enkele puntjes uitgelegd worden voordat we aan de effectieve uitleg kunnen beginnen. Eerst en vooral moeten we weten wat we willen voorspellen. Aangezien we beginnen met binaire logistische regressie maken we een voorspelling tussen twee klassen. Onze vraag kan dus als volgt luiden "Speelt de gebruiker Mortal Kombat of niet?". De waarde y zal ons het resultaat geven. y kan slechts twee waarden aannemen $y \in \{0, 1\}$ met



Figuur 2.1: Sigmoïd functie, wanneer $\theta^T x_i = 0$ zal de sigmoïd functie gelijk zijn aan 0.5 dit wil zeggen dat de 2 klassen even veel kans hebben om gekozen te zijn. Naar mate $\theta^T x_i$ van waarde verhoogt zal de waarschijnlijkheid voor de positieve klasse ook verhogen.

1 als de positieve klasse, Mortal Kombat. Als y 0 is dan duidt het op de negatieve klasse, in onze data stelt dit Pacman voor.

Logistische regressie start vanuit de hypothese van lineaire regressie die als volgt is:

$$h(x_i) = \theta^T x_i = \theta_0 + \theta_1 x_{i1} + ... + \theta_n x_{in}$$

θ^T is de getransponeerde vector van parameters die door het algoritme gegenereerd worden. Als de uitkomst van deze vergelijking ≥ 0 dan zal de positieve klasse voorspelt worden, de andere klasse zal dan gegeven worden wanneer $h(x) < 0$.

De logistische regressie is niet meer dan de sigmoïd functie van die lineaire hypothese. Zo krijgen we $h(x) = g(\theta^T x)$. De sigmoïd functie wordt ook wel logistische functie genoemd. Vandaar de naam logistische regressie.

$$g(x) = \frac{1}{1+e^{-x}} => h(x) = \frac{1}{1+e^{-\theta^T x}}$$

De eigenschap van een sigmoïd funtie is dat altijd zal voldoen aan volgende voorwaarde: $0 \leq h(x) \leq 1$. Nu weten we nog niet wat de waarde van $h(x)$ precies uitdrukt. Dit kunnen we wiskundig uitdrukken als $P(y = 1|x;\theta)$. M.a.w. de kans dat $y = 1$ voor de gegeven vector x met de parameters θ. Wanneer je de kans op de negatieve klasse wenst te weten moet je eenvoudig weg $1 - P(y = 1|x;\theta)$. x kan nu gemakkelijk geclassificeerd worden. Wanneer $h(x) \geq 0.5$ heeft x de meeste kans om tot de positieve klasse te behoren, voor deze casus Mortal Kombat. Anderzijds als $h(x) < 0.5$ zal x behoren tot de andere klasse.

Multiklasse logistische regressie

Wat u uit de naam al kan afleiden is dat dit een algoritme is om een classificatie te maken over meerdere klassen. Dit kan op twee methoden gedaan worden. De one-vs-one methode of de one-vs-rest(/all) methode.

One-vs-all Dit is de gemakkelijkste van de twee methoden. Zoals de naam al doet vermoeden vergelijken we één klasse tegenover alle andere klassen. Als er n aantal klassen zijn dan zullen er n hypothesen $h^{(k)}$ met $k \in \{pacman, MortalKombat, tetris, ..\}$ gemaakt worden. k is dan de positieve klasse en alle andere klassen samen is dan één negatieve klasse.
Voor een nieuwe input x die moet geclassificeerd worden gaat de methode als volgt te werk. $h^{(k)}(x)$ geeft een waarde terug die de waarschijnlijkheid uitdrukt dat x tot de klasse k behoort. Dit wordt voor alle n hypothesen gedaan. De klasse met de hoogste waarschijnlijkheid wordt dan logischerwijs gekozen als de voorspelde klasse waar x toe behoort.

One-vs-one Met deze techniek vergelijken we twee klassen met elkaar zoals we gezien hebben in sectie 2.2.1 binaire logistische regressie. Er wordt dus opnieuw meerdere hypothesen gemaakt maar anders dan in one-vs-all worden er nu combinaties van twee klassen gebruikt wat er zo uitziet $h^{(k,m)}$. k is in dit geval de positieve klasse en m de negatieve. In totaal zullen er n(n-1)/2 hypothesen gemaakt worden. om te bepalen onder welke klasse een nieuwe input x hoort berekenen we $h^{(k,m)}(x)$ wanneer het resultaat ≥ 0.5 krijgt de positieve klasse (k) een "punt". Dit gebeurt voor alle hypothesen en de klasse met het meest aantal punten zal uiteindelijk het resultaat zijn van het algoritme.

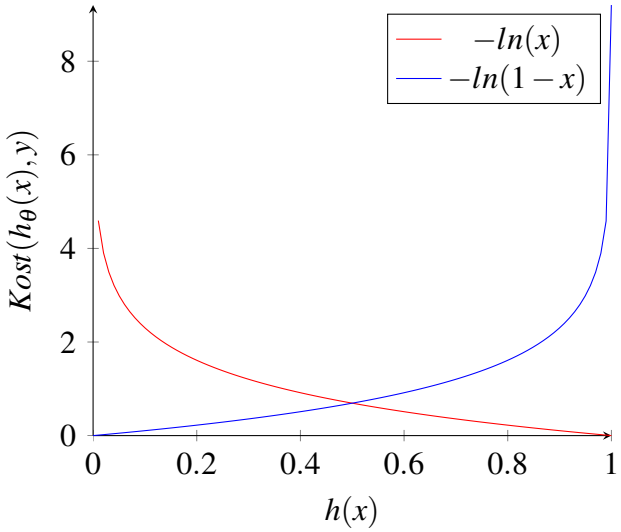
Optimaliseren van algoritme

Om een zo optimaal mogelijk algoritme te verkrijgen moet er gebruik gemaakt worden van optimalisatietechnieken. Gradiënt descent is zo'n techniek die de parameters θ optimaliseert om een zo'n correct mogelijke hypothese te krijgen. Voordat gradient descent besproken wordt gaan we eerst de kostfunctie bespreken.

Kostfunctie logistische regressie

Een kostfunctie wordt genoteerd als $J(\theta)$. Die functie drukt de gemiddelde kost van de trainingsset met parameters θ uit. Aan de hand van gradient descent is het dan mogelijk om de parameters te optimaliseren zodat de kostfunctie geminimaliseerd wordt. Hoe lager de kostfunctie is hoe beter de hypothese. De logistische regressie kostfunctie ziet er als volgt uit:

$$J(\theta) = \frac{1}{m}\sum_{1}^{m} Kost(h_\theta(x^{(i)}), y^{(i)})$$



Figuur 2.2: Visualisatie functies

De kost wordt als volgt berekent:

$$Kost(h_\theta(x),y) \;=\; -y\ln(h_\theta(x)) \;-\; (1-y)\ln(1-h_\theta(x)) \quad met\ y \in \{0,1\}$$

We stellen y gelijk aan 1. Dan zien we dat eigenlijk enkel het eerste deel van de functie $(-y\ln(h_\theta(x)))$ van belang zal zijn want het tweede deel $(-(1-y)\ln(1-h_\theta(x))$ zal nul zijn omdat 1-y dan nul zal zijn en dus het product ook nul is. Zo krijgen we:
$Kost(h_\theta(x),y) \;=\; -\ln(h_\theta(x))$. Op deze manier komen we uit bij de rode kostfunctie die u ziet in figuur 2.2 p17.

$h_\theta(x)$ is de hypothese die uitgelegd is in 2.2.1 binaire logistische regressie. Dus die drukt de waarschijnlijkheid uit dat een vector x positief of negatief is. Wanneer we een perfect voorspelling willen dan zou de kost 0 moeten zijn enkel zo ben je volledig zeker dat het label die aan x wordt toegekend 100% positief of negatief is. Als de hypothese zo goed als zeker is dat het vector positief is zal de kost nog dicht bij nul zijn. Eens $h(x) < 0.5$ zal de kost sneller stijgen. Dit zelfde principe geldt voor de negatieve voorbeelden.

De kostfunctie in één uitdrukking is:

$$J(\theta) \;= -\frac{1}{m}\sum_{1}^{m} y\ln(h_\theta(x)) \;-\; (1-y)\ln(1-h_\theta(x))$$

Er zijn nog andere kostfuncties die gebruikt kunnen worden maar deze wordt altijd Deze functie kan afgeleid worden door een principe binnen statistiek namelijk het "maximum likelihood estimation-principe Dit is een manier om parameters te zoeken voor verschillende modellen zoals logistische regressie. Deze functie is ook convex. Convex is een eigenschap die gebruikt wordt voor optimalisatiefuncties. Dit houdt onder meer in dat de functie het globaal minimum bereikt.

Gradient descent

Gradient descent is een optimalisatiealgoritme die veel gebruikt wordt voor logistische regressie. Op deze manier is het mogelijk om de parameters θ te minimaliseren zodat de hypothese goede voorspellingen kan doen. Gradient descent vereist een continue afleidbare functie, de kostfunctie die we zonet besproken hebben voldoet aan die vereisten. Er is ook een functie g nodig die een vector teruggeeft als gradiënt en een stapgrootte α. De stapgrootte zorgt ervoor dat het algoritme naar een lokaal of globaal minimum kan convergeren. Wanneer het algoritme stopt in een lokaal minimum heb je niet altijd de beste oplossing tenzij het lokaal minimum ook het globaal minimum is. Indien dit niet het geval is kan gradient descent een aantal keer herhaalt worden zodat het globaal minimum kan gevonden worden. Het belang van een goede stapgrootte α is belangrijk. Wanneer die te groot is dan zou het kunnen dat het algoritme in een oneindige lus geraakt. Of als α te klein is dan kan het zeer lang duren tegen dat er een lokaal minimum gevonden is.

Voor iedere parameter θ_n wordt volgende uitdrukking berekent en terug toegekend aan zichzelf

$$\theta_i = \theta_i - \frac{\partial}{\partial}(J(\theta_i, ..., \theta n))$$

(Misschien nog voorbeeldje uitwerken en lokaal/globaal minimum uitleggen)

2.2.2 Support Vector Machine

Andrew Ng is professor aan de Stanford University begon het deel support vectormachine in zijn cursus (Ng, g.d.) met volgende zin.

> SVMs are among the best (and many believe are indeed the best) 'off-the-shelf' supervised learning algorithms.

SVMs scoren over het algemeen beter dan logistische regressie of neurale netwerken. Het is mogelijk dat een van de twee alternatieven beter scoort maar dit hangt dan af van de data die gebruikt wordt.

Support Vector Machines are among the most robust and successful classification algorithms [8, 6]. They are based upon the idea of maximizing the margin i.e. maximizing the minimum distance from the separating hyperplane to the nearest example. The basic SVM supports only binary classification, but extensions [21, 4, 9, 15] have been proposed to handle the multiclass classification case as well. In these extensions, additional parameters and constraints are added to the optimization problem to handle the separation of the different classes. The formulation of [22, 4] can result in a large optimization problem, which may be impractical for a large number of classes. On the other hand, [9] reported a better formulation with a more efficient implementation.

3. Uitwerking

3.1 Benodigdheden

3.1.1 Data

Een onmisbaar onderdeel om aan machine learning te doen is uiteraard de data. Tijdens de eerste fases van het project wordt er gebruik gemaakt van zelf gegenereerde data in Excel. De data die uit de arcademachine ontvangen zal worden zal in CSV-formaat zijn. Doordat Excel-tabbladen opgeslagen kunnen worden in een CSV-file is dit dan ook een logische keuze. Er zullen zo'n 500 metingen beschikbaar zijn die kunnen ingelezen worden in het programma.

We willen uiteraard weten hoe goed het algoritme scoort dit kunnen we testen door testdata te voorzien. De dataset die we uit Excel halen zal opgesplitst worden in trainingsdata en testdata. Het algoritme zal met de trainingsdata een hypothese vormen die ervoor zal zorgen dat ook nieuwe of nog ongekende data een goede voorspelling krijgt. Daarom is het belangrijk om een dataset te hebben die data bevat die nog niet eerder gebruikt is door het algoritme. Enkel op deze manier kunnen we zeker zijn dat het algoritme een goede hypothese heeft gemaakt. Als testdata nemen we 20% van de hele dataset.

3.1.2 Frameworks

Zoals in de inleiding reeds besproken was zijn er verschillende soorten frameworks die reeds geïmplementeerde algoritmen hebben. Een eerste mogelijk framework was TensorFlow van Google. Dit is ontwikkeld in Python, de algoritmen die hierin voorzien zijn, zijn voornamelijk neurale netwerken of deep learning algoritmen. Het is mogelijk om

ook eenvoudigere algoritmen te gebruiken maar daar ligt de specialiteit niet op. En dit in combinatie met een taal die ik niet machtig ben lijkt mij geen goede keuze. Er zijn nog een aantal andere frameworks die gemaakt zijn in andere programmeertalen zoals Javascript, Node.js, Java, etc.

Accord-framework is daar een van, dit is ontwikkeld in .NET. Alle algoritmen die nodig zijn om deze bachelorproef tot een succesvol einde te brengen zijn beschikbaar.
Door te werken met het Accord-framework bestaat er een mogelijk om in de toekomst een webapplicatie te ontwikkelen. Dit openend dus ook nog extra mogelijkheden om te experimenteren met Artificiële Intelligentie.
.NET is een nog niet zo'n populaire taal om aan artificiële intelligentie te doen maar er zit potentieel in. Doordat .NET vele gelijkenissen heeft met Java is er een nog groter publiek die hiermee aan de slag kan.
Door deze mogelijkheden lijkt dit dan ook het meest geschikte framework om aan het werk te gaan.

3.2 Fase 1

De eerste stap om een programma te maken om voorspellingen te doen is de data maken. Omdat dit ook nog maar de eerste fase is beginnen we gemakkelijk. Dit wil zeggen dat we de computer een eerste voorspelling laten doen tussen twee totaal verschillende spelletjes. We nemen Pacman en Mortal Kombat (schietspel) om te starten. Pacman kan gespeeld worden enkel d.m.v. joystickbewegingen. Om Mortal Kombat te spelen heb je veel de knoppen nodig maar ook de joystick. Nu als mens is het gemakkelijk om dit verschil te kunnen zien, als er knoppen gebruikt geweest zijn was de speler Mortal Kombat aan het spelen.

3.2.1 Data

De eerste dataset is zelf gegenereerd zonder echte input van de arcademachine omdat het systeem die de data zal inlezen nog in ontwikkeling is op dit moment. In Excel zijn er drie kolommen voorzien een kolom voor het aantal keer dat de knoppen ingedrukt zijn geweest gedurende twee minuten. Dan het aantal keer dat de joystick bewogen is in diezelfde tijdsspanne. De derde kolom staat het ID van het spel. In figuur 3.1 ziet u random 10 voorbeelden uit de dataset. Het GameID 0 wijst op Pacman en 1 is dan Mortal Kombat. Als u de rijen van Pacman bekijkt dan valt op dat de ButtonPresses niet 0 zijn, dit komt omdat er bij het begin van een spel soms eens geprobeerd wordt wat de functies van de knoppen zijn.

	A	B	C
1	ButtonPresses	JoystickMovements	GameID
2	180	281	1,00
3	2	275	0,00
4	170	449	1,00
5	5	415	0,00
6	1	387	0,00
7	177	375	1,00
8	180	285	1,00
9	10	362	0,00
10	3	245	0,00

Figuur 3.1: Tien voorbeelden van games

3.2.2 Logistische regressie

Het eerste algoritme die we gaan testen is logistische regressie. In fase 1 gaan we slecht voorspellingen doen tussen twee verschillende spellen vandaar dat we binaire logistische regressie gaan toepassen die eerder uitgelegd is in sectie 2.2.1.

In de code 3.2 ziet u de implementatie van de binaire logistische regressie. Er worden 2 parameter meegegeven in de functie, input en output. De input is een dubbele array van *double* waarden. Daarin zitten rijen met 2 kolommen die de ButtonPresses en JoystickMovements bevatten. De ouput parameter bevat een enkele array die het GameID bevat. Hoe deze data ingelezen wordt kan u zien in de code 3.3. Met een ExcelReader krijgen we een DataTable die we met de methode ToJagged kunnen omvormen naar ons gewenste dubbele array met de kolomnamen kunnen we de kolommen selecteren. Deze methode is onderdeel van het Accord-framework.

Wat de LearningRate precies inhoudt wordt later samen met gradient descent uitgelegd (gradient descent zal in het theoretische deel uitgelegd worden)

Resultaten

Snelheid van het algoritme Een eerste belangrijke metric om algortimen te kunnen vergelijken is de snelheid. Het zou niet correct zijn om een stopwatch te laten lopen bij het begin van het programma en te stoppen wanneer het programma klaar is. De tijd om de data in te laden in het programma, de omzetting naar array's, de objecten die worden geïnitialiseerd, ... is allemaal niet zo belangrijk. Wat wel interessant is, is de tijd die het algoritme nodig heeft om tot een hypothese te komen. Dit gebeurt in de *Learn* methode. Net voor die lijn code wordt uitgevoerd starten we een stopwatch en erna stoppen we en bekijken het verschil. Omdat de duur verschillend kan zijn nemen we het gemiddelde van 50 metingen. Doordat deze dataset slechts 400 voorbeelden bevat en de verschillen tussen de spelletjes Pacman en Mortal Kombat zeer duidelijk zijn heeft het algoritme slecht

```
public static void StartLogisticRegression(double[][]
   input, int[] output)
{
        //LogisticRegression object initialiseren met 2
           inputparameters (ButtonPresses &
           JoystickMovements)
        LogisticRegression logisticRegression = new
           LogisticRegression()
        {
                NumberOfInputs = 2
        };

       var learner = new
          LogisticGradientDescent(logisticRegression)
       {
                LearningRate = 0.0001
       };

      // De gradient descent begint met de logistische
         regressie te optimaliseren
      logisticRegression = learner.Learn(input, output);
}
```

Code 3.2: Implementatie binaire logistische regressie

1,53842 milliseconden nodig om een hypothese te maken.

F-score (Moet nog theoretisch uitgelegd worden bij methodologie)
Doordat deze dataset zo simpel is en de testdataset dus ook heeft het algoritme geen probleem om de testdata te classificeren. De precisie is dan ook logischerwijs één en de rappel eveneens. Zo bekomen we een precisie van 1 die dus wil zeggen dat het algoritme foutloos is op de testdata.

Als ik meer data zou voorhanden hebben kan ik de McNeymar's test doen maar in deze fase en deze hoeveelheid data is die test niet echt handig

```
using (var table = new
   ExcelReader("../../datasetExcel.xls").GetWorksheet("Training"))
   {
        // Convert the DataTable to input and output
           vectors
        double[][] inputs =
           table.ToJagged<double>("ButtonPresses",
           "Joystickmovements");
        int[] outputs =
           table.Columns["Game"].ToArray<int>();

        ...
   }
```

Code 3.3: Inlezen Excel data

4. Conclusie

Curabitur nunc magna, posuere eget, venenatis eu, vehicula ac, velit. Aenean ornare, massa a accumsan pulvinar, quam lorem laoreet purus, eu sodales magna risus molestie lorem. Nunc erat velit, hendrerit quis, malesuada ut, aliquam vitae, wisi. Sed posuere. Suspendisse ipsum arcu, scelerisque nec, aliquam eu, molestie tincidunt, justo. Phasellus iaculis. Sed posuere lorem non ipsum. Pellentesque dapibus. Suspendisse quam libero, laoreet a, tincidunt eget, consequat at, est. Nullam ut lectus non enim consequat facilisis. Mauris leo. Quisque pede ligula, auctor vel, pellentesque vel, posuere id, turpis. Cras ipsum sem, cursus et, facilisis ut, tempus euismod, quam. Suspendisse tristique dolor eu orci. Mauris mattis. Aenean semper. Vivamus tortor magna, facilisis id, varius mattis, hendrerit in, justo. Integer purus.

Vivamus adipiscing. Curabitur imperdiet tempus turpis. Vivamus sapien dolor, congue venenatis, euismod eget, porta rhoncus, magna. Proin condimentum pretium enim. Fusce fringilla, libero et venenatis facilisis, eros enim cursus arcu, vitae facilisis odio augue vitae orci. Aliquam varius nibh ut odio. Sed condimentum condimentum nunc. Pellentesque eget massa. Pellentesque quis mauris. Donec ut ligula ac pede pulvinar lobortis. Pellentesque euismod. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent elit. Ut laoreet ornare est. Phasellus gravida vulputate nulla. Donec sit amet arcu ut sem tempor malesuada. Praesent hendrerit augue in urna. Proin enim ante, ornare vel, consequat ut, blandit in, justo. Donec felis elit, dignissim sed, sagittis ut, ullamcorper a, nulla. Aenean pharetra vulputate odio.

Quisque enim. Proin velit neque, tristique eu, eleifend eget, vestibulum nec, lacus. Vivamus odio. Duis odio urna, vehicula in, elementum aliquam, aliquet laoreet, tellus. Sed velit. Sed vel mi ac elit aliquet interdum. Etiam sapien neque, convallis et, aliquet vel, auctor non, arcu. Aliquam suscipit aliquam lectus. Proin tincidunt magna sed wisi. Integer blandit

lacus ut lorem. Sed luctus justo sed enim.

Morbi malesuada hendrerit dui. Nunc mauris leo, dapibus sit amet, vestibulum et, commodo id, est. Pellentesque purus. Pellentesque tristique, nunc ac pulvinar adipiscing, justo eros consequat lectus, sit amet posuere lectus neque vel augue. Cras consectetuer libero ac eros. Ut eget massa. Fusce sit amet enim eleifend sem dictum auctor. In eget risus luctus wisi convallis pulvinar. Vivamus sapien risus, tempor in, viverra in, aliquet pellentesque, eros. Aliquam euismod libero a sem.

Nunc velit augue, scelerisque dignissim, lobortis et, aliquam in, risus. In eu eros. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Curabitur vulputate elit viverra augue. Mauris fringilla, tortor sit amet malesuada mollis, sapien mi dapibus odio, ac imperdiet ligula enim eget nisl. Quisque vitae pede a pede aliquet suscipit. Phasellus tellus pede, viverra vestibulum, gravida id, laoreet in, justo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer commodo luctus lectus. Mauris justo. Duis varius eros. Sed quam. Cras lacus eros, rutrum eget, varius quis, convallis iaculis, velit. Mauris imperdiet, metus at tristique venenatis, purus neque pellentesque mauris, a ultrices elit lacus nec tortor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent malesuada. Nam lacus lectus, auctor sit amet, malesuada vel, elementum eget, metus. Duis neque pede, facilisis eget, egestas elementum, nonummy id, neque.

Bibliografie

Creeger, M. (2009). CTO Roundtable: Cloud Computing. *Communications of the ACM*, *52*(8), 50–56.

Knuth, D. E. (1998). *The art of computer programming, volume 3: (2nd ed.) sorting and searching*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc.

Ng, A. (g.d.). *CS 229: Machine Learning*. Stanford University.

Accord framework. (g.d.). Verkregen van www.accord-framework.net

Tensorflow framework. (g.d.). Verkregen van www.tensorflow.org

Pollefliet, L. (2011). *Schrijven van verslag tot eindwerk: do's en don'ts*. Gent: Academia Press.

Lijst van figuren

2.1 Sigmoïd functie, wanneer $\theta^T x_i = 0$ zal de sigmoïd functie gelijk zijn aan 0.5 dit wil zeggen dat de 2 klassen even veel kans hebben om gekozen te zijn. Naar mate $\theta^T x_i$ van waarde verhoogt zal de waarschijnlijkheid voor de positieve klasse ook verhogen. 15

2.2 Visualisatie functies 17

3.1 Tien voorbeelden van games 21

3.2 Implementatie binaire logistische regressie 22

3.3 Inlezen Excel data 23

Lijst van tabellen