Kan Kunstmatige Intelligentie als Onderwerp van een Lessenserie het Inzicht van Toepassing van Wiskunde verbeteren?

Ruben Snijders

Augustus 2024

Contents

1 Inleiding

Met het opkomen van AI (Artificial Intelligence, oftewel kunstmatige intelligentie) en het veelvuldig gebruik van ChatGPT in de klas zijn leerlingen zich steeds meer bewust van de mogelijkheden van programma's die gebruik maken van kunstmatige intelligentie. Echter, hoewel het gebruik van programma's als ChatGPT wijd verspreid is, blijft het begrip van **hoe** het werkt achter. Dit is een gevaar, aangezien het altijd belangrijk is om te begrijpen hoe de programma's werken waar je gebruik van maakt. Natuurlijk kan je in drie lessen niet direct uitleggen hoe geavanceerde programma's zoals Chat-GPT werken, maar hopelijk legt het wel een basis vanwaaruit leerlingen zelf verder op onderzoek uit kunnen. Het tweede doel van deze lessenserie is meer een algemene interesse opwekken in de leerlingen. Door een hedendaags voorbeeld te geven van hoe wiskunde gebruikt kan worden, krijgen de leerlingen deels antwoord op de eeuwige vraag: "Maar meneer/mevrouw, waar is wiskunde nou goed voor?". Deze lessenserie is ontworpen voor wiskunde B leerlingen uit 5 vwo, bekend met vectoren en het sommatie-teken.

Het onderzoek voorafgaande aan dit verslag bestond uit drie fasen. De eerste fase was het maken van de lessenserie. Hiervoor was het onderwerp onderzocht en verwerkt in drie lessen. De tweede fase bestond uit het beoordelen hoe goed de les voldeed aan verschillende criteria. Om niet afhankelijk te zijn van scholen en hun schema was de keuze gemaakt om de tweede fase te laten bestaan uit feedback van vakdocenten in plaats van de lessen in de praktijk te brengen. Hiervoor is feedback verzameld van vier eerstegraads wiskunde docenten. Deze feedback is vervolgens verwerkt in een nieuwe versie van de lessenserie wat het uiteindelijke product is van dit onderzoek.

Dit proces is in omgekeerd chronologische volgorde verwerkt in dit verslag. Het eerste deel bestaat uit de uiteindelijke lessenserie. Het tweede deel bestaat uit de verantwoording voor deze lessenserie. Dit bevat niet alleen de feedback van de vakdocenten en de daarop gebaseerde veranderingen, maar ook een verklaring van verschillende ontwerpbeslissingen die zijn gemaakt gebaseerd op verschillende literaire bronnen. De oude versie van de lessenserie, waar de feedback van de docenten op gebaseerd is, is bijgevoegd in de appendix. Hier is ook de letterlijke feedback van de docenten gegeven.

De lessenserie zelf bestaat uit drie lessen. De eerste les behandelt regressie problemen (problemen waarbij een getal voorspeld wordt), de tweede les classificatie problemen (waarbij dingen in groepen verdeeld worden) en de laatste les gaat over de verschillende parameters die zijn geïntroduceerd in de eerste twee lessen en hun rol in het leerproces van de AI. De lessenserie wordt afgesloten met een wiskunde cheat sheet, waar bepaalde regels opnieuw worden benoemd, en een docentenhandeleiding voor de drie lessen, bestaande uit extra achtergrond informatie over de onderwerpen die worden besproken in de les, dingen om extra aandacht aan te besteden als de les gegeven wordt en een geschatte tijdsplanning van 50 minuten waarin de les gegeven kan worden.

Part I
Nieuwe versie

2 Leerling werkbladen

2.1 Les 1: Regressie

2.1.1 Inleiding

Deze komende twee lessen zullen we behandelen hoe wiskunde die jullie de afgelopen jaren geleerd hebben gebruikt kan worden om computers iets te laten leren. Hier zullen jullie als leerlingen eerst de rol spelen van een computer en de stappen doorlopen om een voorspelling te maken. Daarna wordt uitgelegd hoe een AI (Artificial Intelligence, oftwel Kunstmatige Intelligentie) dit automatisch kan doen en op veel ingewikkeldere problemen.

2.1.2 Wat is Regressie?

Deze les focust zich op een bepaald soort probleem: regressie. Dit zijn problemen waar het doel is om een continu getal te berekenen gebaseerd op wat eigenschappen. Dit wordt gedaan door een lijn te trekken gebaseerd op de data die de AI heeft, vergelijkbaar met hoe jullie een lineaire functie op kunnen stellen door twee punten. Voorbeelden van mogelijke regressie problemen zijn het voorspellen van beurskoersen, huizenprijzen, hoe veel iemand zal verdienen. Zoals je ziet zijn dit allemaal problemen waarbij je een specifiek getal wil voorspellen. Dit is ten opzichten van de problemen die we de volgende les zullen bespreken waar je dingen in verschillende groepen wil verdelen. Mensen die ook wiskunde A volgen kunnen hierin het verschil herkennen tussen kwalitatief en kwantitatief, waar kwalitatief gaat over getallen en kwantitatief gaat over groepen die niet getallen zijn.

Deze les zullen we eerst handmatig proberen te doen wat we de computer automatisch willen laten doen om dit soort problemen op te lossen. Dat is een lijn trekken die zo dicht mogelijk langs alle punten heen gaat die we hebben. Dit is vergelijkbaar met een lineaire formule opstellen. Daarna gaan we kijken welke formules de computer gebruikt om zo een formule op te stellen. Hiervoor moeten we een formule opstellen voor hoe ver de lijn van de punten afstaat en de afgeleide daarvan zo klein mogelijk maken. Immers kunnen we normaal een minimum vinden door de afgeleide gelijk te stellen aan 0*. Tot slot gebruiken we deze formules op een praktisch voorbeeld om te zien hoe het werkt.

2.1.3 Zelf Proberen

Het voorbeeld waar we deze les mee zullen beginnen is het voorspellen van hoe de zeespiegel zal stijgen in de komende jaren.

Hier zie je dat er ongeveer elk half jaar een meting is gedaan van de zeespiegel. Door een lijn te trekken die zo goed mogelijk overeenkomt met de gemeten data, kan deze lijn doorgetrokken worden om zo voorspellingen te doen over de waterspiegel in de toekomst. Zoals je ziet gaat de lijn niet door alle punten. Het

*Hoewel dat niet direct kan hier, om redenen waar we later wat dieper op ingaan, kunnen we hem wel stapsgewijs zo klein mogelijk proberen te krijgen.

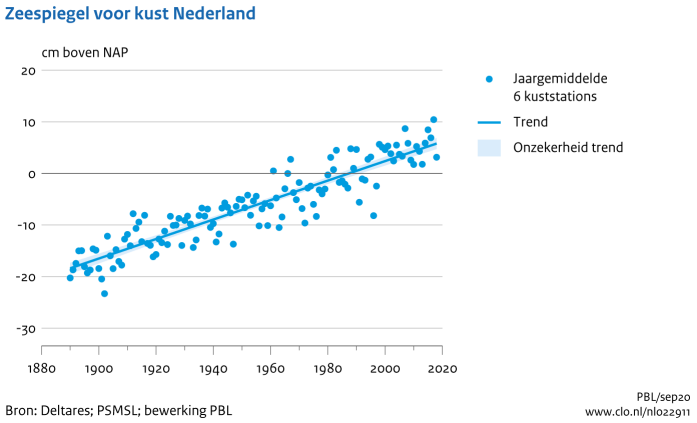

Figure 1: Niveau zeespiegel (CLO 2020)

is vrijwel onmogelijk om zulke dingen exact te voorspellen aangezien er altijd factoren zijn die je niet meeneemt in de meting, al dan niet omdat je ze over het hoofd ziet, of omdat het je model te ingewikkeld maakt. Daarom is het doel om de lijn zo goed mogelijk te trekken aan de hand van de eigenschappen die je hebt. Hoewel het op het eerste gezicht makkelijk lijkt om een redelijke lijn te trekken gebaseerd op de punten, moet deze lijn heel precies getrokken worden om een goede voorspelling te kunnen maken. Als je de lijn maar een beetje scheef trekt, kan je op een paar jaar in de toekomst er opeens toch best ver naast zitten.

Om aan te geven hoe makkelijk het is om er een beetje naast te zitten en een verkeerde voorspelling te maken is de volgende korte opgave. Kijk hiervoor naar grafiek 2 op de volgende pagina.

Hier zijn honderd metingen gedaan en we zien een duidelijk verband tussen de x-as en de y-as. Trek de beste (rechte) lijn die volgens jou het verband aangeeft tussen x en y.

Misschien merk je wel dat het moeilijk is om te bepalen wat de beste lijn is om te trekken. Om te kijken hoe goed de lijn is die je hebt getrokken, meet de hoek tussen de lijn die je hebt getrokken en de x-as. Mogelijk moet je hiervoor jouw lijn en de x-as een beetje naar links doortrekken. Bereken de waarde voor $x = 150$ door $y = tan(\text{hoek}) \cdot 150 + 20$ te berekenen. Als je het precies goed hebt gedaan zou je op 80 uit moeten komen. Als je maar een paar graden ernaast zit met een hoek van 20° of 25°, kom je al op een getal als 74 of 89 uit. Een verschil van ongeveer 10%, wat best significant is. Hierom is het belangrijk dat deze lijnen zo precies mogelijk door het midden van alle punten worden getrokken.

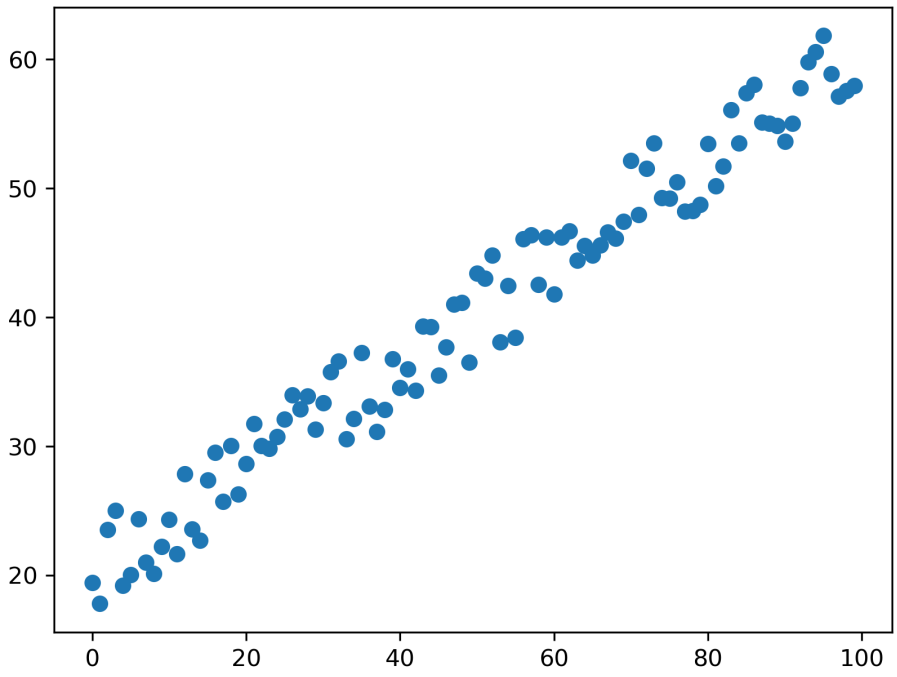

Figure 2: Opgave lijn trekken

2.1.4 Overzicht

Voor we in detail gaan kijken hoe de computer zo een lijn trekt, kijken we eerst naar hoe alle stappen ongeveer werken.

1. De computer begint met een willekeurige gok voor een lijn. Meestal zit deze er enorm naast, maar dat is niet erg.

2. Vervolgens kijkt de computer per meting welk y-coördinaat deze lijn geeft en welk y-coördinaat de meting zegt dat er moet zijn. Denk hierbij aan de vorige opdracht, waar de meting zegt dat je op 80 uit moet komen bij $x = 150$ en jullie getekende lijn bijvoorbeeld op 89 uitkwam. Dit is dus hoe ver de lijn van alle punten afligt. Deze afwijking willen we natuurlijk zo klein mogelijk maken.

3. We zijn dus op zoek naar de formule zodat deze afstand geminimaliseerd is. En als we iets willen minimaliseren gebruiken we natuurlijk de afgeleide. De afgeleide kunnen we niet direct gelijk stellen aan 0, zoals je normaal zou doen, omdat we niet een normale functie hebben. In plaats daarvan gebruiken we de afgeleide als een "richting" waarop we de formule van de lijn moeten veranderen (denk aan een hogere of lagere richtingscoëfficient of startgetal) om deze afgeleide kleiner te maken.

4. Dan verandert de computer de formule van de lijn een klein beetje in die richting en beginnen we weer vanaf stap 1 met deze veranderde lijn als nieuwe gok.

5. Deze stappen blijven herhaald worden tot de lijn goed genoeg is.

2.1.5 Voorspellen

Laten we nu kijken hoe de computer dat daadwerkelijk doet. Zoals je ziet is de eerste stap het trekken van een willekeurige lijn. Dit wordt gedaan in de vorm van een gewichtsvector $\mathbf{g} = (g_s; g_x)$. Om een bijbehorend y-coördinaat te berekenen wordt deze vector vermenigvuldigd met de x-coördinaat van het punt dat we willen voorspellen in de vorm $v = g_x \cdot x + g_s$. Hierin herken je misschien de standaard vorm van een lineaire formule $y = ax + b$, met $a = g_x$ en $b = g_s$. Hier geldt $y = v$, de voorspelling die is gemaakt.

2.1.6 Afwijking Berekenen

We hebben nu gezien hoe een computer een lijn trekt, maar nog niet hoe deze veranderd kan worden om dichter bij alle punten te liggen. Hiervoor moeten we eerst kijken naar stap 2 uit het overzicht, namelijk wat de functie is waarmee we de afwijking tussen de lijn en alle punten kunnen berekenen.

De afwijking (hierna aangegeven met E voor het Engelse "Error") is een functie van de afstand tussen de voorspelling v_i en de werkelijke waarde w_i die y heeft. De $_i$ geeft hier het nummer aan van de x die we willen berekenen. Bijvoorbeeld de eerste meting kan een x hebben van 3 en een werkelijke waarde

y van 5. Dat betekent dus dat $x_1 = 3$ en $w_1 = 5$. De afwijking E die wij gebruiken wordt berekend met de volgende functie:

$$E = \frac{1}{2N} \sum_{i=0}^{N} (v_i - w_i)^2 \tag{1}$$

Wat we hier doen is we kijken dus naar een van de metingen, aangegeven met de i. Hier nemen we het verschil tussen de voorspelling v en de werkelijke waarde w. Dit is de afwijking en wordt berekend met het $(v_i - w_i)$ gedeelte. De volgende stap is een soort speciale versie van het gemiddelde, namelijk de gemiddelde kwadratische afwijking (Mean Squared Error (MSE) in het Engels). Dit speciale gemiddelde wordt berekend door het kwadraat te nemen van deze afwijking, dit te doen met alle metingen, die allemaal bij elkaar op te tellen (aangegeven door het $\sum_{i=0}^{N}$ gedeelte) en dan te delen door het totale aantal metingen (het $\frac{1}{N}$ gedeelte). Dat optellen en delen door het totale aantal is natuurlijk het gemiddelde berekenen. Zoals je ziet wordt er ook nog gedeeld door 2, daarom is het $\frac{1}{2N}$ en niet $\frac{1}{N}$. Dit is om een latere stap makkelijker te maken en daar komen we dan op terug. Nu kan je je afvragen waarom we hier een kwadraat nemen van het verschil. Immers zou je dit normaal niet doen voor een normaal gemiddelde. Neem als voorbeeld een lijn voor de geest die we door vier punten heen willen laten gaan. Bij twee punten gaat hij 5 eronder langs en bij twee andere punten 5 erboven. Bedenk zelf wat deze formule voor E aangeeft als we het kwadraat weghalen en wat als we het wel erbij zetten[†].

2.1.7 Vector Aanpassen

Nu we een formule hebben om te zien hoe goed onze lijn is en hoe ver deze van alle punten af ligt, kunnen we kijken hoe deze lijn dichter bij de punten gelegd wordt. Dit wordt gedaan door de gewichtsvector van de lijn aan te passen met de volgende formules:

$$g_{s_1} = g_{s_0} - \eta \frac{\delta E}{\delta g_{s_0}} \tag{2}$$

$$g_{x_1} = g_{x_0} - \eta \frac{\delta E}{\delta g_{x_0}} \tag{3}$$

Hier zie je dus hoe de coördinaten van de gewichtsvector van de eerste gok voor de lijn (g_{s_0}, g_{x_0}) worden aangepast naar de coördinaten van de tweede gok voor de lijn (g_{s_1}, g_{x_1}). De $_0$ geeft hier aan dat dit de eerste gok is die de computer heeft gedaan. De tweede gok wordt aangegeven met een $_1$ et cetera[‡]. Merk op

[†]Als het goed is kom je zonder kwadraat op een E van 0 uit, wat natuurlijk helemaal niet de bedoeling is. Dat zou alleen maar moeten gebeuren als de lijn precies door alle punten gaat. Zonder het kwadraat valt een afwijking naar boven bij een punt weg tegen een afwijking naar beneden bij een ander punt. Door het kwadraat toe te voegen maakt het niet meer uit of de afwijking naar boven of naar beneden is omdat het "-"teken wegvalt door het kwadraat.

[‡]Er wordt begonnen met een 0 in plaats van met een 1 omdat computers altijd beginnen te tellen met 0

dat hier elke coördinaat van de gewichtsvector apart wordt aangepast gebaseerd op de afzonderlijke afgeleiden.

Laten we eerst kijken naar deze formule en waarom deze werkt. We willen ervoor zorgen dat E zo klein mogelijk is. Oftewel wanneer E minimaal is. Als je het minimum wil vinden, zet je normaal de afgeleide naar 0. Helaas kan dit niet zo makkelijk. Denk zelf na waarom dat is wanner we later deze afgeleide van E uit gaan schrijven[§].

Omdat we niet direct de optimale oplossing kunnen vinden door de afgeleide gelijk te stellen aan 0, gebruiken we de afgeleide op een andere manier. Namelijk als een soort richting waarop we moeten bewegen om het minimum te vinden[¶]. De afgeleide geeft dus de richting aan waarin we g gaan veranderen. Door deze afgeleide af te trekken van het getal wat we al hadden, bewegen we richting het minimum. η geeft dan hoe groot de stap is die we in die richting gaan zetten. Dit heet de leersnelheid. Instinctief wil je deze leersnelheid groot maken, want dan kom je logischerwijs sneller bij het minimum. Het probleem hiermee is dat als je leersnelheid te groot is, je zo ver over je minimum heen stapt dat je op een grotere afwijking uitkomt dan waar je begon. Voor deze lessen hoef je je daar geen druk over te maken en staat per opdracht erbij hoe groot η moet zijn.

De afgeleide $\frac{\delta E}{\delta g_{x_0}}$ kan je als volgt berekenen door de formule voor E, die we hebben gezien bij 1, in te vullen. Hieronder staat deze afgeleide uitgewerkt.

- Van stap 1 naar 2 wordt de formule voor E ingevuld.
- Van stap 2 naar stap 3 wordt het deel van de formule voor E die niet afhankelijk is van g_{x_0} buiten de afgeleide gezet [‖].
- Van stap 3 naar stap 4 gebruiken we de kettingregel om $v_i - w_i$ te vervangen voor u.
- van stap 4 tot stap 7 wordt het $\frac{\delta u}{\delta g_{x_0}}$ deel uitgewerkt. Eerst door u in te vullen, dan door v_i in te vullen en tot slot door de afgeleide uit te werken.
- In de laatste stappen wordt het $\frac{\delta(u^2)}{\delta u}$ deel uitgewerkt en alles zo klein mogelijk geschreven.

[§]Deze afgeleide kan alleen maar 0 zijn als de afstand voor elk punt 0 is, wat niet handmatig te doen is door één keer de lijn te trekken.

[¶]Daarom dat de afgeleide ook 0 is op het minimum of maximum, aangezien je dan er al bent en je niet meer hoeft te bewegen om het minimum/maximum te vinden.

[‖] vergelijkbaar met hoe je als je $3x^2$ afleidt, je x^2 eerst afleidt naar $2x$ en dan aan het eind pas vermenigvuldigt met die 3. Alles wat hier buiten de afgeleide wordt gezet is eigenlijk die 3

$$
\begin{aligned}
&\frac{\delta E}{\delta g_{x_0}} \\
=&\frac{\delta(\frac{1}{2N}\sum_{i=0}^{N}(v_i - w_i)^2)}{\delta g_{x_0}} \\
=&\frac{1}{2N}\sum_{i=0}^{N}(\frac{\delta((v_i - w_i)^2)}{\delta g_{x_0}}) \\
=&\frac{1}{2N}\sum_{i=0}^{N}(\frac{\delta(u^2)}{\delta u}\cdot\frac{\delta u}{\delta g_{x_0}}) \\
=&\frac{1}{2N}\sum_{i=0}^{N}(\frac{\delta(u^2)}{\delta u}\cdot\frac{\delta(v_i - w_i)}{\delta g_{x_0}}) \\
=&\frac{1}{2N}\sum_{i=0}^{N}(\frac{\delta(u^2)}{\delta u}\cdot\frac{\delta(g_{s_0} + g_{x_0}\cdot x_i - w_i)}{\delta g_{x_0}}) \\
=&\frac{1}{2N}\sum_{i=0}^{N}(\frac{\delta(u^2)}{\delta u}\cdot x_i) \\
=&\frac{1}{2N}\sum_{i=0}^{N}(2u\cdot x_i) \\
=&\frac{1}{2N}\sum_{i=0}^{N}(2(v_i - w_i)\cdot x_i) \\
=&\frac{1}{N}\sum_{i=0}^{N}((v_i - w_i)\cdot x_i) \\
=&\sum_{i=0}^{N}\frac{(v_i - w_i)\cdot x_i}{N}
\end{aligned}
$$

Op dezelfde manier kan de afgeleide $\frac{\delta E}{\delta g_{s_0}}$ uitgeschreven worden:

$$
\begin{aligned}
&\frac{\delta E}{\delta g_{s_0}} \\
=&\frac{\delta(\frac{1}{2N}\sum_{i=0}^{N}(v_i - w_i)^2)}{\delta g_{s_0}} \\
=&\frac{1}{2N}\sum_{i=0}^{N}(\frac{\delta((v_i - w_i)^2)}{\delta g_{s_0}}) \\
=&\frac{1}{2N}\sum_{i=0}^{N}(\frac{\delta(u^2)}{\delta u}\cdot\frac{\delta u}{\delta g_{s_0}}) \\
=&\frac{1}{2N}\sum_{i=0}^{N}(\frac{\delta(u^2)}{\delta u}\cdot\frac{\delta(v_i - w_i)}{\delta g_{s_0}}) \\
=&\frac{1}{2N}\sum_{i=0}^{N}(\frac{\delta(u^2)}{\delta u}\cdot\frac{\delta(g_{s_0} + g_{x_0}\cdot x_i - w_i)}{\delta g_{s_0}}) \\
=&\frac{1}{2N}\sum_{i=0}^{N}(\frac{\delta(u^2)}{\delta u}\cdot 1) \\
=&\frac{1}{2N}\sum_{i=0}^{N}2u \\
=&\frac{1}{2N}\sum_{i=0}^{N}2(v_i - w_i) \\
=&\frac{1}{N}\sum_{i=0}^{N}v_i - w_i \\
=&\sum_{i=0}^{N}\frac{v_i - w_i}{N}
\end{aligned}
$$

Wat er nu overblijft is dat het x- en s-coördinaat van de gewichtsvector wordt aangepast met het gemiddelde van hoe ver de lijn die de AI trekt naast elk punt ligt (vermenigvuldigd met de x-coördinaten van elk punt).

Hier zien we nu eindelijk terug wat het nut was van het delen door 2 in formule 1. Als we dat daar niet hadden gedaan, kwamen we bij deze aanpassingsformule twee keer zo hoog uit. Nu zou dat niet heel veel uitmaken, maar dat is minder netjes.

Nu hebben we dus als uitgeschreven aanpassingsfunctie

$$g_{s_1} = g_{s_0} - \sum_{i=0}^{N} \eta \frac{v_i - w_i}{N} \tag{4}$$

$$g_{x_1} = g_{x_0} - \sum_{i=0}^{N} \eta \frac{(v_i - w_i) \cdot x_i}{N} \tag{5}$$

2.1.8 Opdracht 1

Om het effect van deze verandering te laten zien kijken we naar de volgende dataset. Dit is de dataset van verschillende leerlingen met hoeveel uur ze leren voor een bepaalde toets. Dit gaan we gebruiken om te voorspellen wat voor cijfer deze leerlingen gaan halen. Hier gaan we er even van uit dat het behaalde cijfer alleen afhankelijk is van hoe lang de leerling leert. Hieronder staan in tabel 1 30 leerlingen. De eerste kolom geeft elke leerling een nummer, de tweede geeft aan hoeveel uur de leerling leert, en de laatste kolom geeft de werkelijke waarde aan met het cijfer dat die leerling haalt. Deze data is ook weergegeven in grafiek 3.

Leerlingnummer	Aantal uur leren (x)	Behaalde cijfer (y)
0	0.5	2.3
1	1	2.7
2	2	3.5
3	2	3.5
4	2.5	3/9
5	3	4.3
6	3	4.3
7	3.5	4.7
8	3.5	4.7
9	4	5.1
10	4	5.1
11	4	5.1
12	4.5	5.5
13	4.5	5.5
14	4.5	5.5
15	5	5.9
16	5	5.9
17	5	5.9
18	5.5	6.3
19	5.5	6.3
20	5.5	6.3
21	6	6.7
22	6	6.7
23	6.5	7.1
24	7	7.5
25	7.5	7.9
26	8	8.3
27	9	9.1
28	9.5	9.5
29	10	9.9

Table 1: Data voor het regressie probleem

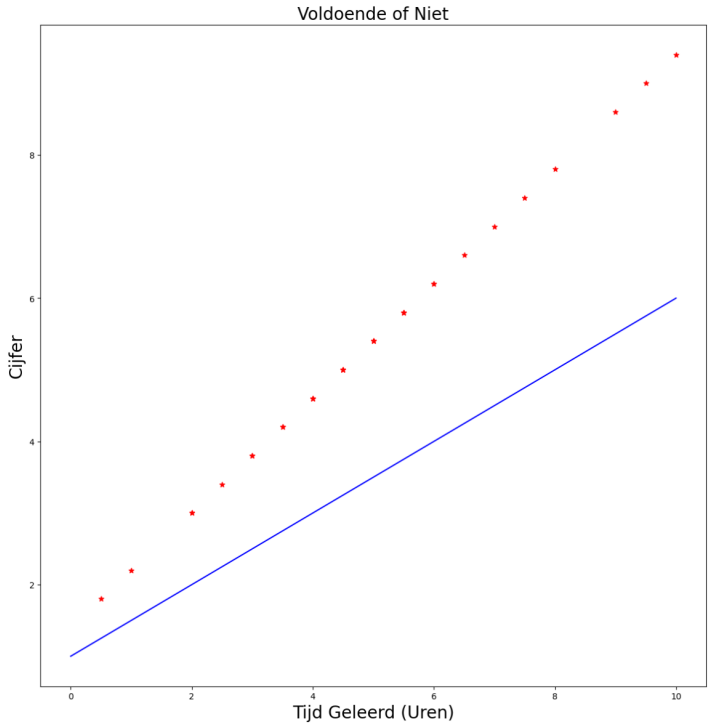

Figure 3: Grafiek met de behaalde cijfers en de begin vector van de computer

Gegeven is de gewichtsvector aan het begin $\mathbf{g} = (1; 0, 5)$ en de leersnelheid η van 0.001.

Gebruik nu de functies 4 en 5 om deze vector aan te passen. Verdeel de metingen over de klas zodat iedereen apart de aanpassing door een van de metingen berekent. Tel die allemaal bij elkaar op en kijk hoe de nieuwe grafiek zal lopen. Als jullie het allemaal goed hebben gedaan komen jullie als het goed is uit op een vector van ongeveer $(1, 07; 0, 87)$. Deze loopt als de groene lijn in figuur 4 op de volgende pagina.

Zoals je ziet is dit in een keer al een stuk beter, maar nog niet zo goed als het zou kunnen zijn. Als deze aanpassingen meerdere keren worden gedaan zal de lijn steeds dichter bij de meetpunten komen. Zodra de lijn zo goed is als we hem willen hebben zou je de gewichtsfactor kunnen gebruiken om, met het aantal uur dat een leerling leert voor de toets, te kunnen berekenen wat voor cijfer die leerling haalt.

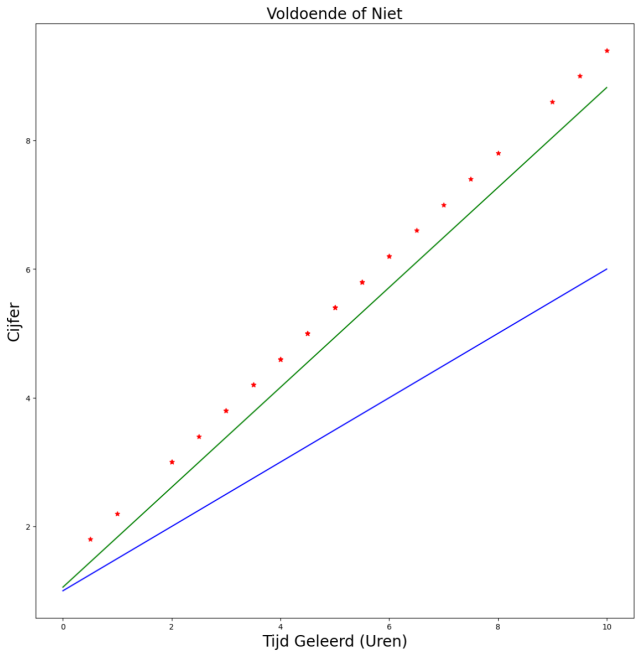

Figure 4: De aangepaste gewichtsvector

2.1.9 Conclusie

Nu kan je je indenken dat je prima handmatig een lijn hier doorheen zou kunnen trekken en zelfs de bijpassende formule kunnen maken. Het voordeel van het op deze manier doen is dat het de computer niet uitmaakt hoe veel eigenschappen we gebruiken. Zo kunnen we bijvoorbeeld ook meten hoe goed een leerling is in een vak met hun gemiddelde cijfer, of het cijfer op de vorige toets. Daarnaast zijn er nog andere dingen waarvan het afhangt hoe goed je het doet op een toets, zoals hoe goed je hebt geslapen, welke dag het is, hoe laat de toets is. Dat zijn allemaal eigenschappen die we gewoon aan dit model toe kunnen voegen om het cijfer mee te berekenen, maar met al deze eigenschappen kunnen we grafiek niet meer zomaar op papier zetten.

Daarnaast is het verband tussen het cijfer dat is gehaald en hoe lang er geleerd is nu lineair. De computer kan ook ingewikkeldere verbanden gebruiken, zoals bijvoorbeeld een wortel- of gebroken verband om aan te geven dat meer leren niet altijd beter is. Het verschil tussen 1 of 2 uur leren is immers veel groter dan tussen 10 en 12 uur. Als je de juiste verbanden weet te vinden (of ingewikkeldere AI maakt die zelf verschillende verbanden uit kan proberen) en genoeg informatie erin stopt kan je bijna alles wel voorspellen.

2.2 Les 2: Classificatie

2.2.1 Inleiding

Vorige les werd er gefocust op regressie problemen. Dat waren problemen waarbij het doel was om een specifiek getal te voorspellen door een lijn te trekken die zo dicht mogelijk bij de metingen ligt. Natuurlijk willen we ook andere dingen kunnen herkennen/voorspellen die niet afhankelijk zijn van getallen. Dat maakt het geen regressie-, maar een classificatie probleem.

2.2.2 Wat is Classificatie?

Classificatie problemen zijn problemen waar een voorspelling wordt gedaan of iets tot een bepaalde groep behoort of niet. Dit is terug te zien aan de term "class" in classificatie, wat groep betekent. Het doel van AI in een classificatie probleem is het scheiden van groepen met behulp van lijnen. Voorbeelden van dit soort problemen zijn het herkennen van planten door ze in groepen te verdelen gebaseerd op kenmerken van de planten zoals de vorm van de bladeren en het indelen van gebruikers van bijvoorbeeld Netflix gebaseerd op wat ze kijken om films en series aan te raden; dit is slimmer dan alleen dingen in dezelfde genres aanraden.

Deze les volgt dezelfde opbouw als de vorige les. Eerst beginnen we met het handmatig trekken van lijnen om te doen wat we met de computer willen automatiseren. Vervolgens kijken we weer naar de formules om de computer deze lijnen te laten trekken. We eindigen weer met een praktisch probleem om te zien hoe deze formules werken.

2.2.3 Zelf Proberen

Voordat we gaan kijken hoe een AI lijnen trekt om deze groepen te verdelen, gaan we eerst zelf proberen dit met de hand te doen. Neem de grafiek met de lengte van de werkers en de lengte van de koningin van bijen en wespen. Trek nu een rechte lijn om deze twee van elkaar te scheiden.

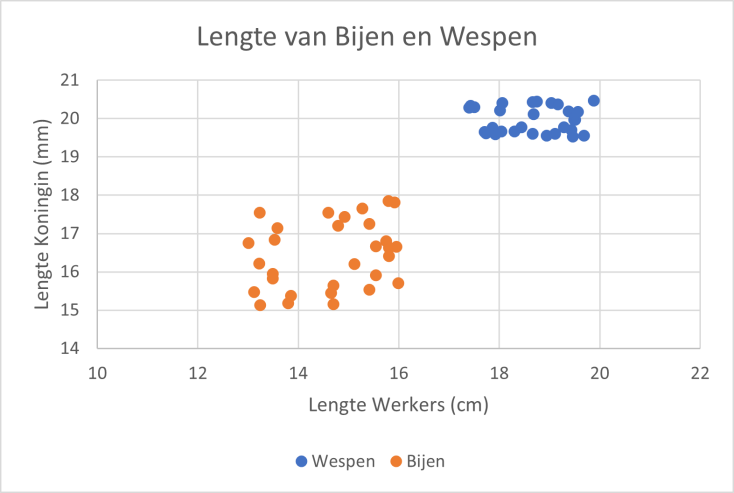

Figure 5: Puntenwolk van korven wespen en korven bijen

Zoals je ziet is dit niet al te moeilijk. Het is zelfs mogelijk om heel veel verschillende rechte lijnen te trekken die nog steeds alle punten goed verdelen.** Echter is het niet altijd zo makkelijk. Probeer nu hetzelfde te doen voor het voorbeeld met het gewicht en de lengte van spechten en roeken.

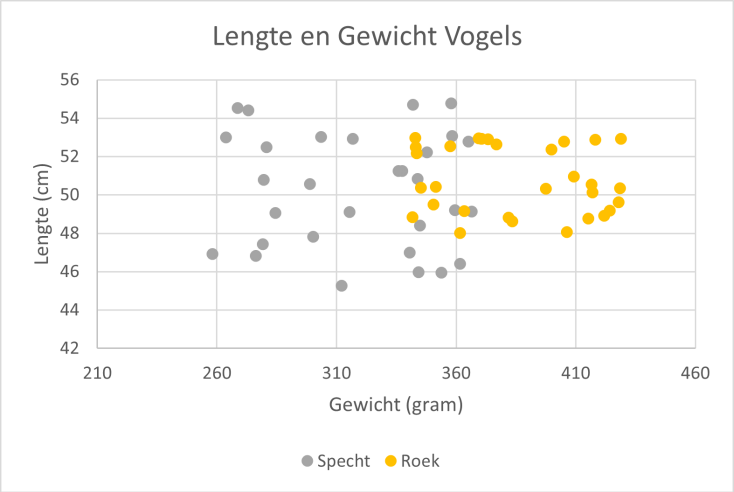

Figure 6: Puntenwolk van de lengte en het gewicht van spechten en roeken

Dit is een stuk moeilijker. Het is zelfs onmogelijk om een rechte lijn te trekken die beide groepen perfect scheidt. In dit geval is het doel om zo min mogelijk punten aan de verkeerde kant van de lijn te hebben. Zoals je ziet is het handmatig wel goed te doen om een lijn te trekken tussen twee groepen die gedefinieerd zijn door twee eigenschappen. Echter zijn heel veel dingen niet op deze manier te classificeren. Sommige dingen zijn bijvoorbeeld afhankelijk van meer eigenschappen dan je makkelijk in een grafiek kan zetten, waardoor het niet zo makkelijk handmatig te classificeren is. Op dezelfde manier zou je niet zo makkelijk handmatig een lijn kunnen trekken als je een vergelijkbaar probleem had als de spechten en roeken, maar dan met duizenden punten in plaats van maar 60. Met 60 kan je nog wel zien wanneer je het minste aantal punten aan de verkeerde kant van de lijn hebt staan, maar met duizenden wordt dat al snel veel moeilijker. Hierom is het gebruik van AI ontwikkeld om goede classificaties te maken.

2.2.4 Overzicht

We beginnen eerst met een algemeen overzicht van alle stappen die de computer neemt om een lijn te trekken om de twee groepen te verdelen. Deze stappen lijken heel erg op de stappen van de vorige les.

**Er zijn manieren om de beste lijn te bepalen in zo een geval. Hiervoor wordt de afstand tussen de lijn en het dichtstbijzijnde punt van elke groep gemaximaliseerd.

1. Wederom beginnen we met een willekeurige lijn als eerste gok.

2. Nu nemen we alle punten en kijken per punt of het aan de juiste kant van de lijn ligt of niet.

3. In tegenstelling tot regressie problemen, hebben nu alleen de punten die aan de verkeerde kant van de lijn liggen effect op hoe we de lijn veranderen. We nemen namelijk de coördinaten van de punten die aan de verkeerde kant van de lijn liggen en bewegen de lijn een stuk in de richting van deze verkeerde punten. Op deze manier komt de lijn aan de andere kant van deze punten te liggen, en komen deze, hopelijk, aan de juiste kant van de lijn te liggen zonder al te veel van de goede punten aan de verkeerde kant te leggen.

4. Met deze nieuwe lijn beginnen we weer bij stap 1 en gaan zo door tot bijna alle punten aan de juiste kant van de lijn liggen.

2.2.5 Vector Opstellen

Laten we nu kijken hoe een AI automatisch een goede lijn kan trekken om de twee groepen te scheiden. De AI begint met een gok voor deze scheidingslijn. Deze lijn is weergegeven door de vector $\mathbf{g}_0 = (g_{s_0}; g_{x_0}; g_{y_0})$. Deze vector vormt de lijn $g_s + g_x \cdot x + g_y \cdot y = 0$, oftewel $y = -\frac{g_x}{g_y} \cdot x - \frac{g_s}{g_y}$. Vervolgens kijkt de AI naar alle datapunten en maakt een voorspelling over deze datapunten. In ons geval wordt dit gedaan door de vector van de lijn ($\mathbf{g}_0$) te vermenigvuldigen met de vector van de eigenschappen van het eerste datapunt $(1; x; y)$. De 1 hier wordt hier toegevoegd voor de vermenigvuldiging met g_s, die als startgetal niet afhankelijk is van x of y. Deze berekening komt op hetzelfde neer als $g_s + g_x \cdot x + g_y \cdot y$. Door de vector van de lijn te vermenigvuldigen met de vector van een punt kunnen we kijken of dit punt boven of onder de lijn ligt[††]. Als de uitkomst positief is wordt voorspeld dat het datapunt boven de lijn ligt en tot de 1 groep behoord en als de uitkomst negatief is onder de lijn en dus tot de -1 groep. Deze uitkomst wordt vergeleken met de werkelijke waarde die bij dit datapunt hoort (w_1) en, als dit niet overeenkomt, beweegt de AI de lijn in de richting van de foute datapunten, zodat die dichter bij de andere, juiste, kant komen te liggen.

Dit wordt gedaan met de volgende formule:

[††]De precieze uitkomst van deze vermenigvuldiging is 2x hoe ver het punt boven of onder de lijn ligt. Als er bijvoorbeeld 4 uit deze vermenigvuldiging komt, dan ligt het punt 2 boven de lijn.

$$g_{s_1} = g_{s_0} + \eta \sum_{i=0}^{N} (1 \cdot g_{s_0}) \tag{6}$$

$$g_{x_1} = g_{x_0} + \eta \sum_{i=0}^{N} (x_i \cdot g_{x_0}) \tag{7}$$

$$g_{y_1} = g_{y_0} + \eta \sum_{i=0}^{N} (y_i \cdot g_{y_0}) \tag{8}$$

Om dit iets duidelijker te maken volgen jullie nu een praktisch voorbeeld:

2.2.6 Opdracht 2

Of leerlingen een voldoende gaan halen op een toets in een toetsweek is afhankelijk van een hele hoop factoren. Dit zijn onder andere hoeveel tijd een leerling heeft geleerd, hoe goed een leerling is in een vak, hoe druk een leerling is. Nu zijn de laatste twee hiervan niet makkelijk numeriek weer te geven, maar dit kan gedaan worden door te kijken naar gevolgen, zoals het vorige cijfer en het gemiddelde cijfer voor hoe goed een leerling in een vak is en het aantal toetsen die een leerling die toetsweek heeft voor hoe druk hij is. In de volgende dataset (tabel 2) is van verschillende leerlingen aangegeven hoe lang zij hebben geleerd voor een toets op de x-as en hoe goed ze staan voor het vak op de y-as. Deze staan ook in grafiek vorm.[‡‡] Daarnaast is ook gegeven of deze leerling een voldoende heeft gehaald of niet in de tabel in kolom w en in de grafiek met de kleuren groen voor voldoende en rood voor onvoldoende.

De gewichtsvector aan het begin is $\mathbf{g}_0 = (-100; 4; 10)$. Dit komt dus overeen met de lijn $-100 + 4x + 10y = 0$ Dit is erg arbitrair, maar werkt goed met een update voor deze les. In les 3 kunnen jullie experimenteren met andere vectoren als begin vector. η voor dit probleem is $0,05$. Vermenigvuldig de gewichtsvector met de coördinaten van de datapunten om de output te berekenen van elk van deze datapunten. Hiervoor vul je de coördinaten van de datapunten dus in in de formule $v = -100 + 4x + 10y$. Als er een positieve waarde uitkomt wordt het datapunt geclassificeerd als een 1, oftewel voldoende, en voor een negatieve waarde een -1, oftewel een onvoldoende. Merk op dat er nu gecontroleerd kan worden of een datapunt wel of niet goed geclassificeerd is berekend kan worden door $(g_{s_i} + g_{x_i} x_i + g_{y_i} y_i) w_i$ te berekenen. Als hier het namelijk als voldoende wordt geclassificeerd en het ook een voldoende is, dan is $(g_{s_i} + g_{x_i} x_i + g_{y_i} y_i)$ positief en $w_i = 1$, dus $(g_{s_i} + g_{x_i} x_i + g_{y_i} y_i) w_i > 0$. Andersom geldt hetzelfde, als het als onvoldoende wordt geclassificeerd en ook onvoldoende is, dan is $(g_{s_i} + g_{x_i} x_i + g_{y_i} y_i)$ negatief en $w_i = -1$, dus weer geldt $(g_{s_i} + g_{x_i} x_i + g_{y_i} y_i) w_i > 0$. Op dezelfde manier, als het verkeerd geclassificeerd is, dan is een van de twee

[‡‡] Merk op dat, ondanks het feit dat de waardes bij x en y numeriek zijn, en we voldoende aangeven als een 1 en een onvoldoende aangeven als een -1, het nog steeds een classificatie probleem is.

Leerlingnummer	Tijd Geleerd (x)	Gemiddeld Cijfer (y)	Wel of geen voldoende (w)
0	0.5	5.7	-1
1	1	7.5	1
2	2	4.1	-1
3	2	5.9	-1
4	2.5	5.7	-1
5	3	2	-1
6	3	5.4	-1
7	3.5	7.5	1
8	3.5	8	1
9	4	6	1
10	4	6.9	1
11	4	3.5	-1
12	4.5	5.4	-1
13	4.5	6.6	1
14	4.5	6.5	1
15	5	5.2	-1
16	5	7.2	1
17	5	8	1
18	5.5	7.3	1
19	5.5	8.5	1
20	5.5	5.5	1
21	6	5.3	-1
22	6	8	1
23	6.5	6.7	1
24	7	7.9	1
25	7.5	7.8	1
26	8	8.2	1
27	9	7.8	1
28	9.5	9	1
29	10	9.1	1

Table 2: Tabel voor het classificatie probleem

negatief en de ander positief, dus is $(g_{s_i} + g_{x_i}x_i + g_{y_i}y_i)w_i < 0$. Voor jullie is het niet direct nodig om het op deze manier te berekenen, aangezien je het punt in de grafiek op zou kunnen zoeken en/of makkelijk zelf kan vergelijken of je uitkomst overeenkomt met wat het moet zijn, maar voor een AI is het wel makkelijk om zo te kunnen berekenen welke er goed zijn en welke niet.

Als alle waardes zijn berekend, update de gewichtsvector aan de hand van formules 6, 7 en 8, met $\eta = 0,05$ en kijk hoe dit de lijn veranderd. Gebruik hiervoor dus alleen de punten die nu verkeerd staan.

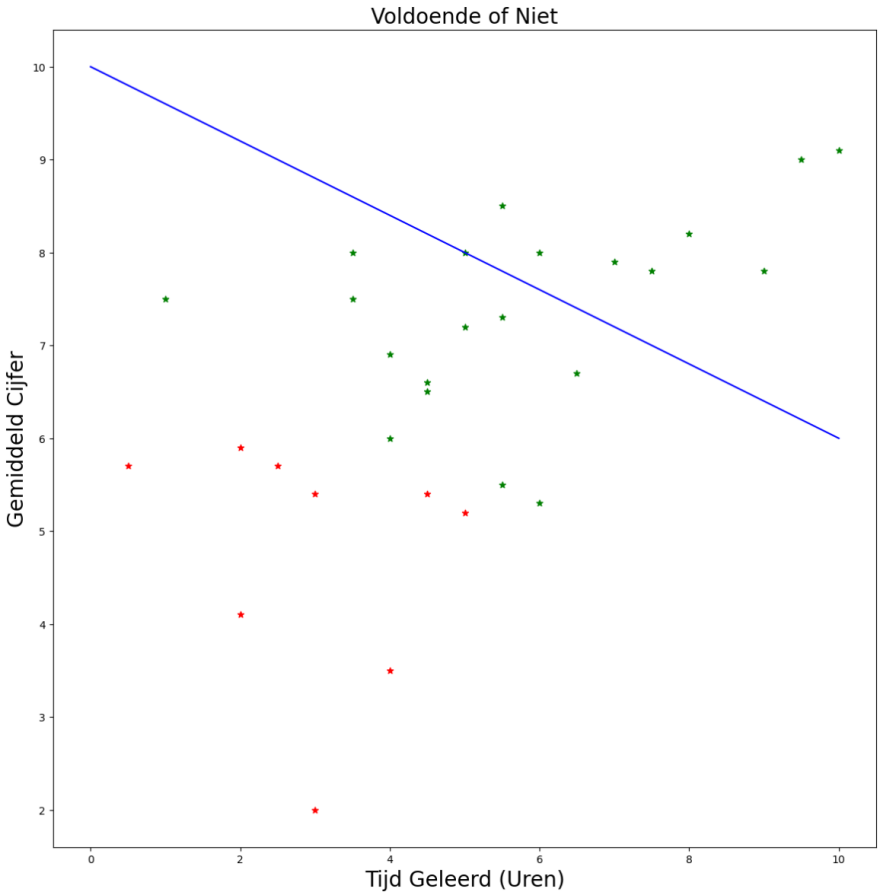

Figure 7: Grafiek voor het classificatie probleem

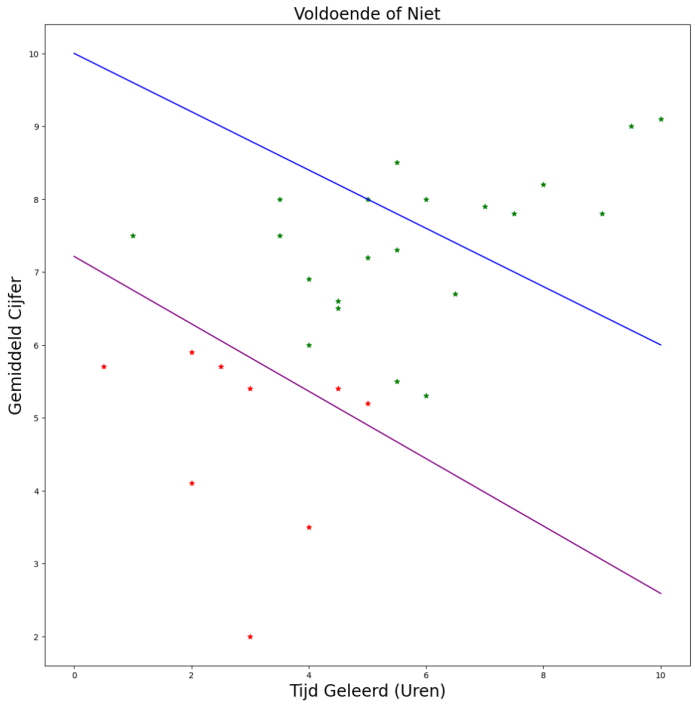

Figure 8: Grafiek voor het classificatie probleem na een update

Als jullie dit als klas goed hebben gedaan is de uiteindelijke nieuwe gewichtsvector $(-99,45; 6,375; 13,785)$. Deze staat in het paars in grafiek 14:

Zoals je ziet is de lijn na een ronde veranderingen al een stuk beter, maar nog niet perfect. Ik vermoed dat na een paar rondes de lijn precies de onvoldoendes van de voldoendes zal scheiden, waarna de gewichtsvector niet meer veranderd omdat er geen foute punten meer zijn.

2.2.7 Conclusie

Net als het regressie probleem is dit een heel simpel probleem waar je als goed nadenkend mens prima zelf een lijn doorheen kan trekken. Het voordeel van deze manier is dat de computer makkelijk meer variabelen mee kan nemen in de berekening (terwijl wij maar 2 kunnen gebruiken als we het nog steeds op papier willen kunnen zien). Daarbovenop werkt deze methode nog steeds om de beste lijn te trekken als er geen perfecte lijn te trekken is omdat de punten meer door elkaar staan, en kan moeilijkere verbanden gebruiken om een niet-rechte lijn te trekken als die beter zou zijn.

2.3 Les 3

Deze laatste les wordt er teruggekeken naar het probleem van afgelopen lessen met het voorspellen van het cijfer dat leerlingen gaan halen. In plaats van een half uur met 30 leerlingen om een enkele update uit te voeren op de gewichtsvector, zullen wij nu code gebruiken om tientallen updates per seconde uit te kunnen voeren en op deze manier snel een optimale uitkomst te berekenen.

Jullie hebben een complete code gekregen en gaan zelf onderzoeken wat het effect is van bepaalde eigenschappen in het programmeren van de vergelijkingen die jullie in de afgelopen lessen geleerd hebben. Dezelfde dataset die jullie gebruikt hebben wordt hier weer gebruikt om de gewichtsvector op te trainen.

Er zijn een paar veel voorkomende fouten die op kunnen komen bij het trainen van een AI. De eerste is een die waarschijnlijk veel voor gaat komen in de code die jullie voor deze les zullen gebruiken. Dit is een probleem van uitschieten. Dit probleem ligt vooral in het kiezen van een te groot getal voor leersnelheid η. Hier is in les 1 al wat over gezegd, maar het doel van η is aangeven hoe snel je de lijn in de juiste richting wil verplaatsen. Instinctief wil je dit getal natuurlijk niet te klein maken, aangezien het dan heel lang duurt voor je de lijn op de juiste plek hebt. Maar hierin schuilt juist dit grotere gevaar van uitschieten.

Neem het volgende voorbeeld van opdracht 2. Zoals je misschien nog weet was η daar 0,05. Nu gaan we dit nog een keer proberen, maar dan met een η van 1. Het resultaat hiervan staat in figuur 9. Hier zien we dat de lijn na een verandering wel de juiste richting is gegaan, maar te ver, waardoor hij nu te laag ligt. Meer lager dan dat de lijn eerst te hoog lag zelfs. Wat er nu zal gebeuren is dat hij bij de volgende ronde nog hoger te hoog komt dan dat hij begon, en de ronde daarna nog meer te laag. Zo blijft hij heen en weer jojoën totdat hij oneindig hoog en/of oneindig laag terecht komt.

Een tweede probleem is het probleem van overfitten. Dit zal waarschijnlijk niet voorkomen met de simpele problemen die wij gebruiken voor deze lessen, maar een heel fundamentele stap in AI trainen is om dit probleem tegen te gaan, daarom wordt het hier wel genoemd. Hiervoor kijken we naar een ander regressieprobleem, in figuur 10, maar deze heeft de vorm van een sinusgolf in plaats van een rechte lijn. De groene lijn is waar de punten (met een kleine afwijking) op zijn gebaseerd, en de rode lijn is de beste lijn die de computer kan maken. Hier beginnen we met een heel makkelijk model van een horizontale lijn (linksboven), naar een lineaire functie (rechtsboven), naar een derdemacht functie (linksonder). Dit model wordt dus steeds verder uitgebreid en verder getraind. Tot het deel rechtsonder waar de lijn perfect door alle punten heen gaat. Maar, zoals je wel ziet, de rode lijn van de computer komt helemaal niet meer overeen met de "werkelijkheid" van de groene lijn. Hierom wordt er altijd maar een deel van de dataset gebruikt om de computer de lijn te laten trekken, en een ander deel om uiteindelijk de afwijking te meten om te bepalen hoe goed de lijn daadwerkelijk is.

Denk aan deze twee valkuilen als je zelf gaat experimenteren met de code. Terwijl je dit doet, probeer zelf de volgende vragen te beantwoorden:

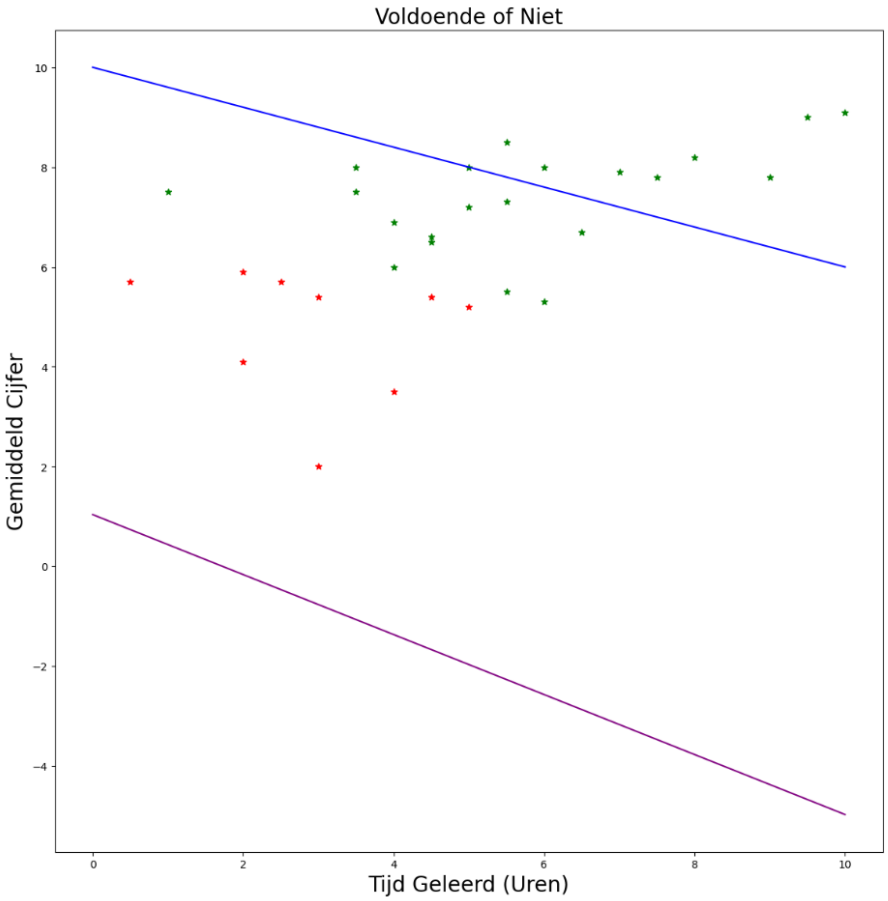

Figure 9: Opdracht 2 met een leersnelheid van 1

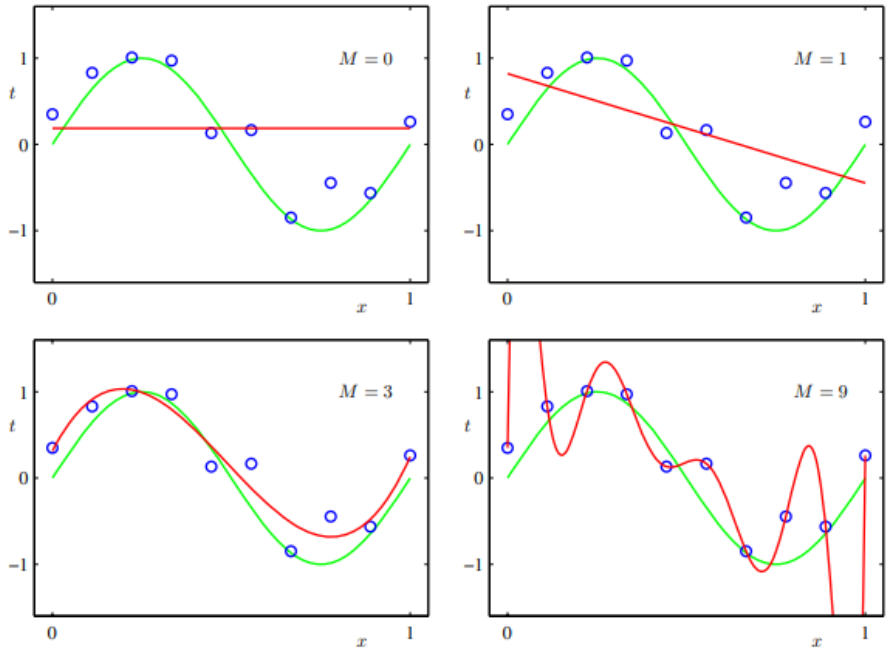

Figure 10: Het risico van het te ver trainen van een code

- Wat voor effect heeft het veranderen van de begin gewichtsvector (W1 voor het regressie probleem en W2 voor het classificatie probleem in de code)?
- Wat voor effect heeft het veranderen van de leersnelheid η?
- Geeft de kleinste MinError ook de beste uitkomst op de nieuwe dataset in het regressie probleem?
- Zijn meer iteraties altijd beter?

2.4 Wiskunde cheat sheet

Dit is een cheat sheet voor jullie om te gebruiken in de volgende lessen. Hier staat welke letter waar voor staat, wat het betekent als een letter vet gedrukt is of als er een klein nummertje rechts onder een letter staat, al dan niet met haakjes eromheen. Daarnaast is er ook een korte uitleg hoe vectoren ook al weer met elkaar werden opgeteld, hoe een vector wordt vermenigvuldigd met een constante en wat regels voor afgeleiden.

2.4.1 Definities

In de volgende lessen zullen wij de volgende wiskundige definities en notaties gebruiken:

- Regressie problemen zijn problemen waarbij een specifiek getal voorspeld moet worden. Hierbij is het doel om de computer een lijn door alle punten heen te laten trekken.
- Classificatie problemen zijn problemen waarbij dingen in groepen moeten worden verdeeld. Hier is het doel om de computer de lijn tussen de twee groepen door te laten trekken, met de ene groep aan de ene kant en de andere groep aan de andere kant van de lijn.
- $\mathbf{g}$ geeft de gewichtsvector aan. Deze is als volgt opgedeeld: $\mathbf{g} = (g_s; g_x; g_y)$. $\mathbf{g}$ wordt meerdere keren aangepast door de AI terwijl het zoekt naar de beste oplossing. Dit wordt aangegeven door een klein getal. Dit ziet er als volgt uit: $\mathbf{g}_2$ is de gewichtsvector nadat die twee keer is aangepast. $\mathbf{g}_2$ bestaat uit $(g_{s_2}; g_{x_2}; g_{y_2})$, waar g_{s_2}, g_{x_2}, en g_{y_2} gewoon getallen zijn, bijvoorbeeld $(1; 2; -3)$.
- $(x, y)_m$ zijn de eigenschappen van de m'de meting. Dus in les 1, $(14; 17)_3$ betekent dat de 3e korf die is gemeten werkers heeft die 14 millimeter lang zijn en een koningin die 17 millimeter lang is.
- Een voorspelling die wordt gemaakt wordt aangegeven door de letter v. Net als met de eigenschappen wordt het nummer van de voorspelling aangegeven met een v^m. Dus om weer hetzelfde voorbeeld te gebruiken, v_3 is de voorspelling die de AI maakt over de 3e korf.
- Een voorspelling wordt gedaan door de eigenschappen $(x; y)$ te vermenigvuldigen met de gewichtsvector $\mathbf{g}$. Dit wordt gedaan door de vector van de eigenschappen (met een 1 ervoor) te vermenigvuldigen met de gewichtsvector. Dus $(g_s; g_x; g_y)$ vermenigvuldigd met $(1, x, y)$ van een datapunt. De voorspelling van korf 3 na de gewichtsvector twee keer aan te passen is bijvoorbeeld $v_3 = g_{s_2} + g_{x_2} \cdot x_3 + g_{y_2} \cdot y_3 = 1 + 2 \cdot 14 + -3 \cdot 17 = 1 + 24 - 51 = -26$.
- Het daadwerkelijke label of getal dat hoort bij de meting wordt aangegeven met de letter w. w_3 is dus het werkelijke label dat hoort bij korf 3.

3 Docenten Handleiding

3.1 Algemeen

Deze lessenserie bestaat uit drie lessen, regressie, classificatie en programmeren. De wiskunde die nodig is in de drie lessen is vectoren, zowel het aflezen als het vermenigvuldigen hiervan, en het begrijpen van formules in de vorm $a + bx + cy = 0$. Het vermenigvuldigen van vectoren wordt expliciet uitgewerkt in de werkbladen, maar als de leerlingen dit al beheersen is het makkelijker voor hen om de rest te begrijpen. Daarnaast wordt er ook gebruik gemaakt van het sommatie teken.

Lessen 1 en 2 introduceren beide een bepaalde vorm van voorspellen. Dit zijn regressie en classificatie respectievelijk. Regressie om een specifieke numerieke waarde bij een datapunt te kunnen voorspellen, classificatie om datapunten in groepen te kunnen verdelen. Denk hierbij aan het verschil tussen kwantitatief en kwalitatief. Hier later meer over bij de uitleg van de specifieke lessen. De leerlingen krijgen die twee lessen eerst een voorbeeld van het soort voorspellen dat er gedaan wordt en spelen zelf de rol van AI door voorspellingslijnen in grafieken te trekken. Hierna worden de leerlingen geïntroduceerd aan de formules die horen bij het leerproces van de AI. Deze formules worden toegepast op een dataset die de leerlingen krijgen. Voor efficiënter werk kan deze dataset verdeeld worden onder de leerlingen, met een datapunt per leerling. Hierna kunnen de leerlingen met hun buren bespreken en elkaars uitkomst controleren. Elke leerling berekend op deze manier de aanpassing die toegepast wordt op de voorspellingslijn. Als alle veranderingen opgeteld worden kan er gevisualiseerd worden wat voor effect deze verandering heeft op de voospellingslijn met het bijgevoegde python bestand.

3.2 Coderen Les 1: Regressie

3.2.1 Opening

Het is handig om de eerste les te beginnen met input uit de klas over waar zij aan denken bij AI en Machine Learning. Persoonlijk vind ik een woordwolk met bijvoorbeeld Wooclap een goede manier om dit te doen aangezien dit ook weergeeft als een antwoord vaak gegeven wordt. Hiermee kan de voorkennis van de leerlingen geactiveerd worden en kan een gesprek gestart worden over de invloed van AI op de hedendaagse samenleving in de vorm van bijvoorbeeld ChatGPT en Dall-E. Natuurlijk gaat de les niet zo ver dat ze zelf een nieuwe versie van deze programma's kunnen maken, maar wel de principes die hier ten grondslag aan liggen.

3.2.2 Achtergrond informatie

Na deze inleiding begint de uitleg met het verschil tussen classificatie en regressie. Dit kan klassikaal, maar staat ook in het werkblad uitgewerkt. Als je dit klassikaal wil bespreken kan dit ook. Hiervoor de volgende uitleg: Classificatie wordt gebruikt om AI dingen in groepen te laten verdelen. Een goed voorbeeld hiervan is bijvoorbeeld Dall-E. Dall-E is een code die kunst kan namaken in de stijl van bekende schilders. Hiervoor moet de computer eerst onderscheid leren maken tussen deze verschillende schilders. Dit wordt gedaan door een AI te laten trainen op schilderijen van bijvoorbeeld Rembrandt en Van Gogh. Nu verdeelt de AI deze gebaseerd op verschillende eigenschappen en probeert Rembrandt van Van Gogh te scheiden gebaseerd op deze eigenschappen. Vervolgens kan de code dan omgedraaid worden om deze eigenschappen te gebruiken om kunst te maken. Regressie wordt gebruikt om een lijn door punten heen te trekken. Dit is vergelijkbaar met opdrachten waar je de formule van een lijn door twee punten heen moet trekken. De computer kan dit doen met meer punten en ook als ze niet allemaal op een rechte lijn liggen. Dit wordt gedaan om specifieke getallen te voorspellen. Denk hierbij aan het voorspellen van wat de marktprijzen zullen zijn of hoe veel regen er zal vallen op een bepaald moment. Als er leerlingen zijn die (ook) wiskunde A volgen, kan de vergelijking gemaakt worden met kwalitatief en kwantitatief. Voor classificatie heb je te maken met een kwalitatieve variabele (zoals dus verschillende schilders) en voor regressie heb je te maken met een kwantitatieve variabele (zoals waterpijl, geld of hoeveelheid neerslag).

3.2.3 Lessamenvatting

Deze les staat in het teken van regressie. Eerst krijgen de leerlingen een voorbeeld van een regressie probleem met een uitleg van het belang van dit soort problemen. Dit wordt gevolgd met een opgave als praktisch voorbeeld waar dit soort regressie problemen voor gebruikt kunnen worden en om duidelijk te maken hoe makkelijk het is om een fout te maken als je dit met de hand probeert op te lossen. Hierna volgt er eerst een globaal overzicht van de stappen die de

computer zet om een zo goed mogelijke regressie lijn te trekken. Dit overzicht is zo simpel en abstract mogelijk gehouden, zodat de leerlingen een referentiekader hebben voor de meer specifieke termen, formules en vectoren die zullen volgen (Wasserman 2015). Na dit overzicht volgen namelijk deze stappen nog een keer, maar dan uitgebreid en gedetailleerd met formules. Voordat de leerlingen aan de klassikale opdracht beginnen kan het handig zijn om even gezamenlijk stil te staan bij 2.1.6. Hier wordt het moeilijkste stukje wiskunde van de dag uitgewerkt en is tegelijkertijd het belangrijkste deel van de les, aangezien dit is hoe de computer de lijn daadwerkelijk zo dicht mogelijk bij de punten krijgt. Als leerlingen hier moeite mee hebben, kan het handig zijn om hier even stil te staan om dit te verbinden aan de wiskunde die de leerlingen al gehad hebben en hoe dat hier naar voren komt (van Ast et al. 2021) (Vygotski 1978). Als eerste wordt er begonnen met hoe de afwijking wordt gedefinieerd. Dit is als de helft van het gemiddelde van het kwadraat van het verschil tussen de voorspelling en het punt zelf. Voor de aanpassing zelf wordt elk coördinaat van de gewichtsvector aangepast met de partiële afgeleide van deze afwijking naar dat coördinaat van de gewichtsvector. Deze aanpassing wordt nog vermenigvuldigd met de leersnelheid η zodat deze niet te drastisch is en overschiet. Deze η wordt later bij 3.2.5 extra uitleg over gegeven, mocht jij hier zelf geïnteresseerd in zijn of als je een nieuwsgierige leerling verwacht. De partiële afgeleide is al voor de leerlingen uitgewerkt, maar het kan fijn zijn om even te controleren dat iedereen door heeft wat er hier gebeurt. Alle stappen die worden gezet zouden stappen moeten zijn die de leerlingen zelf ook kunnen, alleen hebben ze het waarschijnlijk nog niet op deze schaal hoeven doen (Vygotski 1978). Mogelijk moet het $\sum$ teken ook nog even uitgelegd worden aan de leerlingen.

Nu kunnen de leerlingen aan de slag met opdracht 1. Hier gebruiken de leerlingen de eerder besproken formules om de regressielijn een keer te verplaatsen. Hiervoor is een tabel gegeven met 30 datapunten. Om tijd te besparen kan elke leerling een datapunt gegeven worden, waarmee ze persoonlijk de aanpassing aan de gewichtsvector berekenen. Hiervoor moeten ze eerst berekenen wat de voorspelde waarde v is van hun datapunt door de x uit de tabel in te vullen en te vermenigvuldigen met de gegeven gewichtsvector $(1; 0, 5)$, wat neerkomt op $v = 1+0, 5x$. Die gevonden v vullen ze in in formule 4 en formule 5. Nu willen zij per persoon het $\eta\frac{v_i-w_i}{N}$ en $\eta\frac{(v_i-w_i)\cdot x_i}{N}$ gedeelte berekenen. Als leerlingen eerder klaar zijn dan de rest kunnen ze alvast met hun buren bespreken en controleren of hun berekening klopt (Rahimi et al. 2018). Zodra (bijna) iedereen klaar is kan alles bij elkaar opgeteld worden en afgetrokken van de originele vector coördinaten, zoals er gebeurt in de rest van de formules 4 en 5. Als het goed is, is de uitkomst $g_{s_1} = 1, 07$ en $g_{x_1} = 0, 87$ voor $\mathbf{g}_1 = (1, 07; 0, 87)$. Deze staat al getekend op het blad voor de leerlingen. Als jullie op iets anders uitkomen kan je met het python bestand de lijn tekenen die bij jullie nieuwe gewichtsvector hoort. **Schrijf daarin de vector in de Amerikaanse stijl, (1.07,0.85), dus met een "." om een kommagetal aan te geven en een "," om de coördinaten te scheiden.**
Als dit zou worden gedaan op een computer wordt deze vector nu gebruikt en

opnieuw vergeleken met alle datapunten, net zoals alle leerlingen voor deze opgave hebben gedaan met de $(1; 0, 5)$ vector om de lijn, als het goed is, zo steeds dichter bij alle punten te krijgen.

3.2.4 Tijdverdeling

- 10 minuten: Inleiding van het programma
- 5 minuten: opdracht lijn trekken en berekenen van afwijking
- 10 minuten: stilstaan bij de afwijking formule en de aanpassing regel.
- 15 minuten: opdracht 1 maken en klassikaal bespreken
- 10 minuten: uitlooptijd/extra stilstaan bij de vorm van de afwijkingsformule

3.2.5 Extra Informatie

Als er tijd over is kan er een discussie geopend worden over het belang van de vorm van functie 1 en het gebruik van de afgeleide hiervan. Deze functie staat namelijk niet vast, maar is gewoon gekozen voor dit probleem. De vorm van de afwijking gaat vooral over het gebruik van het kwadraat van het verschil tussen de voorspelling en de gewenste waarde. Hoe hoger de macht, hoe groter de verandering voor grote fouten. Dit is goed in de zin dat de gewichtsvector sneller aangepast wordt, maar slecht met het risico dat deze aanpassing te snel gebeurt. Het belangrijkste is dat deze macht even is (of dat er anders gebruik wordt gemaakt van de absolute waarde van dit verschil). Anders streept een afwijking naar boven bij een punt zich weg tegen een afwijking naar beneden bij een ander punt.

Dit gevaar van de vector te snel aanpassen geldt overigens ook voor het kiezen van η. Als de macht van je afwijking of je η te groot is, loop je het risico een grotere afwijking te creëren met het proberen een afwijking weg te werken. Wat de gewenste uitkomst hiervan is, is weergegeven in afbeelding 11. Hier is horizontaal een dimensie van de gewichtsvector weergegeven, en verticaal de afwijking tussen de voorspelling en de gewenste uitkomst. Idealiter wordt de minimum afwijking met kleine stappen benaderd. De afgeleide geeft aan of de gewichtsvector groter of kleiner moet zijn, en de afwijking met η geeft aan hoe groot deze verandering wordt. Als η klein is, wordt het minimum met kleine stapjes benaderd, vergelijkbaar met de situatie in figuur 11. Natuurlijk wil je niet dat deze stapjes te klein zijn, want dan duurt het voor eeuwig voordat je daadwerkelijk bij dit minimum bent aangekomen. Echter wil je deze η ook niet te groot nemen. Het gevaar daarvan is dat de stappen die je neemt zo groot zijn dat de afwijking die je het minimum voorbij schiet en op een afwijking uitkomt die juist groter is dan de afwijking waar je mee begon. Zie figuur 12 voor een visualisatie hiervan. De leerlingen krijgen zelf aan het eind van de volgende les nog een praktisch voorbeeld hiervan.

Een slimme leerling zou zich af kunnen vragen waarom we niet gewoon de afgeleide gelijk nemen aan nul. En dit zou een goede opmerking zijn. Immers is dat de manier waarop normaal gesproken de afgeleide wordt gebruikt om een minimum te vinden. Echter als je de leerling wijst op wat de afgeleide precies wordt, is dit slechts het verschil tussen de lijn en de gewichtsvector, eventueel vermenigvuldigd met het coördinaat van elk punt. Dit kan alleen nul zijn als of de coördinaten van elk punt precies nul is (wat een triviaal probleem is), of als de lijn van de gewichtsvector precies door elk punt zou gaan (technisch gezien mogelijk, maar niet iets wat je zomaar zou kunnen berekenen). Daarnaast is de afgeleide niet zomaar afhankelijk van x op een manier die we goed kunnen voorspellen wanneer de afgeleide in het geheel nul is. We weten alleen de lokale afgeleide.

In plaats daarvan wordt de afgeleide meer gebruikt als een soort wichelroede. We kijken welke richting wij op worden gewezen door de afgeleide, zetten een stap die kant op en kijken dan weer welke richting we op worden gewezen[§§].

Je kan deze aanpassing zelf visualiseren als volgt: Neem bijvoorbeeld afbeelding 11. Hier hebben we een afwijking die alleen afhankelijk is van een coördinaat W. Op het punt D zal de afgeleide van de afwijking negatief zijn. Immers is het een dalende lijn. Door dit af te trekken van de W waar we ons op dat punt bevinden, krijgen we een hogere waarde voor W en hebben we dus een stap naar rechts gezet, richting de minimum afwijking. Mocht een stap ons per ongeluk naar de rechterkant van het minimum brengen, dan is de situatie omgekeerd. De afgeleide is positief, dus door dit af te trekken van de positie waar we ons op dat punt bevinden zetten we een stap naar links, wederom richting het minimum. Deze methode met het bekijken per coördinaat of het minimum hoger of lager ligt wordt dus als het ware per coördinaat in de gewichtsvector gedaan, om op die manier een universeel minimum te bereiken.

De methode die hier wordt gebruikt heet Gradient Descent. Een logische naam omdat de gradiënt (dat is de partiële afgeleide van de afwijking E naar alle verschillende gewichts-coördinaten) hier wordt gebruikt om als het ware stap voor stap af te dalen naar het minimum (Harrington 2012). Hiervoor gebruiken wij de formule gegeven in (Bishop 2006*a*). Er zijn verschillende methoden om gradient descent toe te passen. Dit zijn batch, stochastic en mini-batch. Batch gradient descent gebruikt de hele "batch" aan datapunten in een keer. Hiermee wordt eerst voor elk punt de aanpassing op de gewichtsvector berekend en dan worden al deze aanpassingen in één keer toegepast op de gewichtsvector om de nieuwe gewichtsvector te berekenen. Bij stochastic gradient descent wordt voor het eerste datapunt de aanpassing op de gewichtsvector berekend, deze meteen toegepast en dan met deze nieuwe gewichtsvector wordt er gekeken naar het tweede datapunt enzovoorts. Mini-batch zit hier tussenin waarbij er telkens naar een paar datapunten gekeken wordt, in plaats van maar één datapunt of alle datapunten in één keer (Ketkar & Moolayil 2021). Elke methode heeft zijn eigen nadelen. Batch kost veel computergeheugen omdat hiervoor elk datapunt

[§§]Mogelijk dat leerlingen meer aangaan op een vergelijking als ender pearls om een fortress te vinden in minecraft, maar dat zal ik je niet aandoen.

tegelijkertijd geladen moet worden en is daarom niet nuttig voor datasets van het formaat die meestal voor AI wordt gebruikt. Stochastic update per punt en heeft daarom last van sterk fluctuerende afstanden/fouten. Hierdoor is deze methode erg langzaam of heeft een grote kans minima te missen omdat de aanpassingen te groot zijn. Mini-batch combineert de twee methoden om zo de nadelen te minimaliseren.

In deze lessenserie gebruiken wij de batch methode omdat op deze manier het makkelijk is om alle leerlingen in een keer aan het werk te zetten.

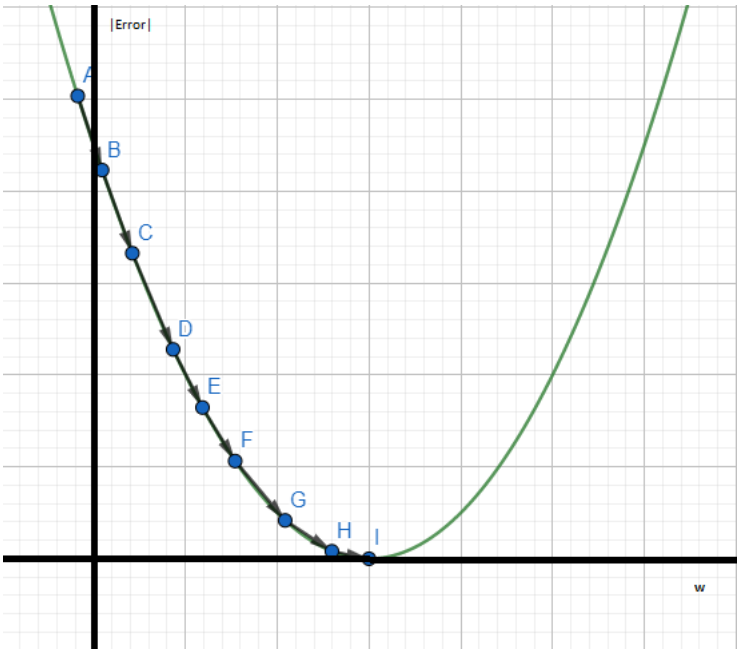

Figure 11: Het gewenste geval

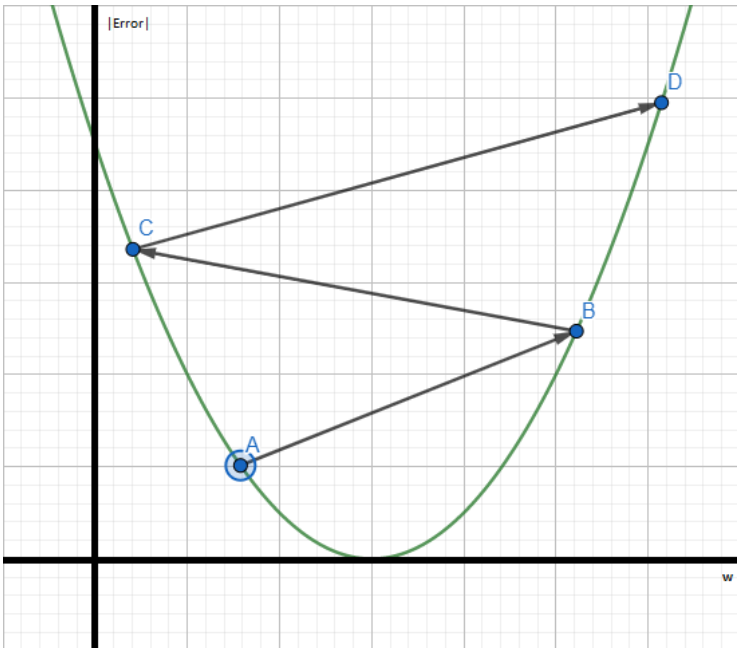

Figure 12: Wat er gebeurt als de gewichtsvector te snel veranderd

3.3 Coderen Les 2: Classificatie

3.3.1 Lessamenvatting

Deze les begint weer met een uitleg wat voor problemen classificatie problemen zijn aan de hand van wat praktische voorbeelden. Hier is het doel dat leerlingen lijnen trekken in grafieken om beide groepen van elkaar te scheiden. Dit is wat de computer probeert te doen en deze kleine opdracht is vergelijkbaar met het trekken van de lijn in les 1.

Zodra de leerlingen een gevoel hebben voor wat het doel is van de classificatie kan je eventueel stilstaan bij waarom de update formules werken. Dit kan eventueel gedaan worden met een overdreven voorbeeld waarbij er 100 datapunten die verkeerd ingedeeld worden op 1 plek zitten en het feit dat de lijn door alle plekken gaat met coördinaten met een verhouding $(a \cdot s; a \cdot x; a \cdot y)$. Als die 100 punten verkeerd geclassificeerd zijn, dan, door 100 keer dat ene datapunt bij de originele weight op te tellen, benader je die verhouding $(a \cdot s; a \cdot x; a \cdot y)$ waardoor de lijn verschoven wordt in de juiste richting.

Tot slot kunnen ze door naar Opdracht 2. Op het formulier van de leerlingen is een dataset gegeven, zowel als grafiek als als tabel. Je kan de leerlingen wederom ieder een punt aangeven, aangezien iedereen 30 punten laten berekenen mij meer tijd lijkt te kosten dan er in de les zit. Hiervan berekenen zij op zichzelf of dit punt goed of fout is geclassificeerd, wederom door de coördinaten van hun punt te vermenigvuldigen met de gegeven vector als volgt: $v = g_s + g_x \cdot x + g_y \cdot y$. Als dit positief is, is $v = 1$ en negatief geeft $v = -1$. Dit wordt vergeleken met de w uit de tabel. Als het punt verkeerd is geclassificeerd moet ook nog, vergelijkbaar met de vorige les, de aanpassing van dat punt berekend worden met het $\eta(1 \cdot w_{s_0})$ gedeelte in formule 6 en de vergelijkbare delen in formule 7 en 8. Hierna kunnen ze in tweetallen hun antwoord bespreken. Tot slot worden alle antwoorden verzameld en kan klassikaal de nieuwe gewichtsvector berekend en bekeken worden. Hieronder is een grafiek toegevoegd waar de nieuwe lijn op aangegeven staat zoals ik die heb berekend met een nieuwe gewichtsvector van $(-99,45; 6,375; 13,785)$ (figuur 14). Als je klassikaal op een andere waarde uitkomt is dit ook te visualiseren met de bijgevoegde code. Let wederom op de manier waarop de vector geschreven moet worden als je dit doet.

3.3.2 Tijdverdeling

- 5 minuten: Inleiding classificatie
- 10 minuten: Opdracht lijnen trekken
- 10 minuten: Stilstaan bij de formule
- 15 minuten: Opgave 1
- 10 minuten: Nabespreken/uitloop

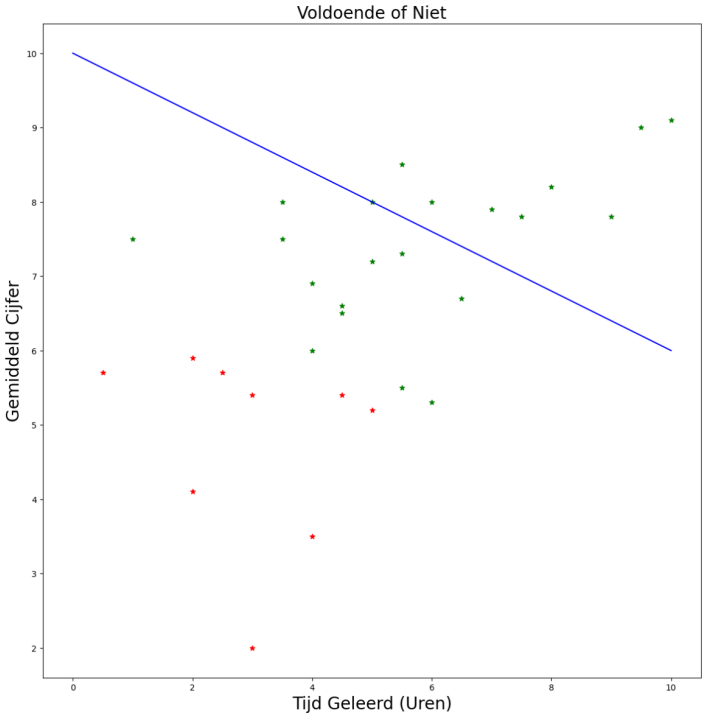

Figure 13: De scheidingslijn aan het begin

3.3.3 Extra Informatie

Voor deze classificatie problemen gebruiken we weer dezelfde methode van Gradient Descent. De standaard formule werkt hetzelfde als de formule die we hebben uit les 1, namelijk formule 2. De afgeleide kan wederom vereenvoudigd worden. In dit geval gebruiken we de formule gegeven door (Bishop 2006*b*). Het effect van deze formule wordt weergegeven in figuur 15. Hier zie je hoe elk punt als het ware trekt aan de lijn om hem zo in de juiste richting te zetten. Merk op dat de aanpassingen hier stochastisch gedaan worden, terwijl we in de les gebruik maken van batch gradient descent, maar het principe blijft hetzelfde.

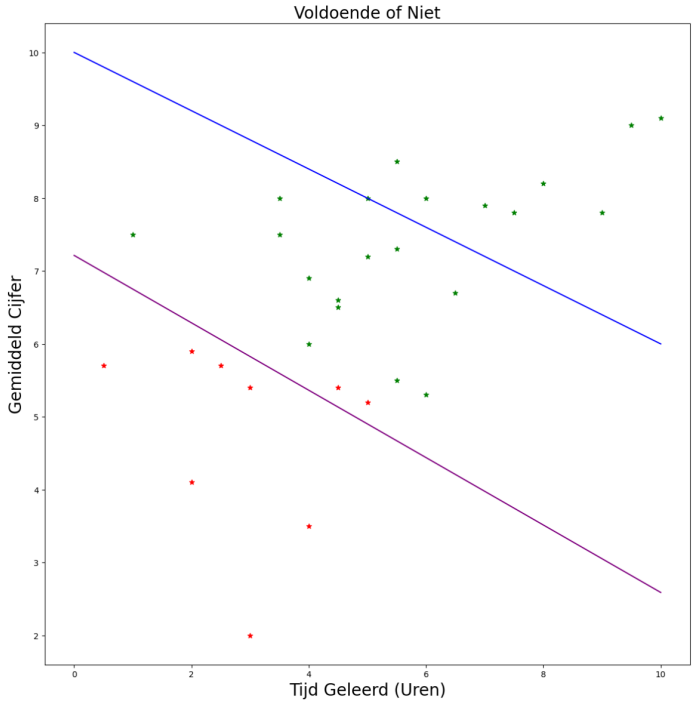

Figure 14: De scheidingslijn na 1 ronde aan veranderingen

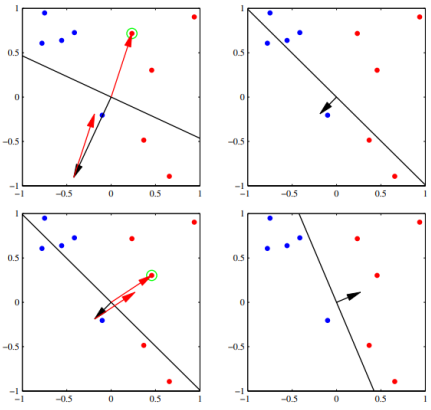

Figure 15: Een visualisatie van de aanpassingen op de schijdingslijn (Bishop 2006*b*)

3.4 Les 3: Programmeren

Deze les begint met een kort stuk over het gevaar van overfitten, de leersnelheid η en hun effect op het leerproces van de AI. Met deze kennis vers in hun geheugen (van Ast et al. 2021) krijgen de leerlingen voor deze les de code die automatisch de gewichtsvector traint op dezelfde data die was gebruikt in de afgelopen lessen. Het doel van de les is dat leerlingen het belang van bepaalde variabelen in het leerproces gaan onderzoeken. Dit is het aantal updates die gedaan worden en de precisie die hiermee bereikt wordt, het effect van de beginwaarde van de gewichtsvector en het effect van de leersnelheid. Het idee is dat de leerlingen hier zelf op onderzoek uitgaan en door trial-and-error op meer inzichten komen dan als het ze gewoon klassikaal wordt uitgelegd. Tevens raad ik aan om de leerlingen hun bevindingen eerst met elkaar te laten bespreken en ze met elkaar eventuele rare uitkomsten uit te laten zoeken voor je ze direct helpt (Schoenfeld et al. 2014) (Rahimi et al. 2018).

De vindingen die ik verwacht dat de leerlingen zullen vinden is dat met een kleinere waardes in de begin gewichtsvector, er sneller een grote afwijking uit komt en dat η kleiner moet worden gemaakt. De gewichtsvector heeft immers kleinere waardes, dus de updates moeten ook kleiner zijn, anders krijg je snel een te grote afwijking. Als leerlingen constant de melding krijgen dat "de afwijking te groot is en dit nergens toe leidt", kan er aangeraden worden om de η kleiner te maken. Dit wordt in de werkelijkheid meestal opgelost door de gewichtsvector en data te normaliseren, maar dat is hier niet ingevoegd. Een andere vinding bij regressie is dat de afwijking bij de test set groter is dan bij de training set. Hoewel dit altijd wel enigszins het geval is kan dit hier een bewijs zijn van overfitten. Natuurlijk wordt, als het goed is, de precisie op de training data ook groter naarmate je meer updates doet. Hopelijk vinden wat leerlingen ook dat met heel veel updates de afwijking op de test data uiteindelijk wel groter wordt, maar de gegeven data en de formules zijn zo simpel dat het niet heel waarschijnlijk is dat dit zal gebeuren.

Tijdverdeling

- 10 minuten: Ophalen van de twee opdrachten en herhalen van het belang van η, het aantal genomen stappen en het gevaar van overfitten.
- 30 minuten: Leerlingen gaan zelf aan de slag met code
- 10 minuten: Nabespreken vindingen van leerlingen

Part II
Verantwoording

4 Verantwoording

Het doel is om leerlingen een nut bij te brengen van de wiskunde die zij leren op de middelbare school. Als onderwerp is gekozen om ze te leren hoe AI werkt. Dit onderwerp is gekozen omdat het een heel actueel onderwerp is dat vaak in het nieuws komt. Dit helpt met de motivatie wekken bij de leerlingen wat op zijn beurt weer helpt met het leerproces (Marzano & Miedema 2018). Een groot probleem bij het programmeren van AI is dat het soms wordt gezien als een black box waar programmeurs wel de code kunnen schrijven, maar niet heel goed alle stappen die hier ten grondslag aan liggen begrijpen. Mijn doel hiermee is leerlingen wat meer relationeel begrip (Skemp 2016) over het onderwerp bij te brengen in plaats van alleen platte regels (Freudenthal 1991). Hierom (en het feit dat dit een wiskunde les is) focussen deze lessen zich op de specifieke stappen die ten grondslag liggen aan het programmeren, in plaats van blindelings programmeren.

Deze lessenserie wordt geopend met het opbrengen van wat de leerlingen al weten als ze denken aan AI. Daarnaast is er een cheat sheet voor de leerlingen met de wiskunde en notatie die gebruikt wordt in de lessen. Dit haalt de voorkennis op en helpt de leerlingen een context te vormen voor wat er geleerd gaat worden (van Ast et al. 2021).

Daarnaast begint elke les, voordat er diep wordt ingegaan op de details en de formules, met een globaal overzicht van alle stappen die zullen volgen. Hiermee hebben de leerlingen een idee van wat er gaat volgen en bereidt het de leerling voor op deze moeilijke stappen (Wasserman 2015).

Deze lessen focussen zich vooral op het gebied waar de leerlingen al bekend mee zijn (afgeleiden en vectoren) zodat de leerlingen een goede kern van kennis hebben om deze nieuwe onderwerpen in de zetten (Vygotski 1978). De opdrachten die de leerlingen in de eerste twee lessen maken zijn ontworpen dat de leerlingen zowel enige productive struggle ondervinden (Schoenfeld et al. 2014), samen kunnen werken als ze er niet uit kunnen komen en hun eigen stappen uit moeten leggen om beter de te maken stappen te begrijpen en tot slot klassikaal om nog steeds te controleren of de antwoorden goed zijn. Deze lesmethode is gebaseerd op de Denken-Delen-Uitwisselen methode (Rahimi et al. 2018).

De laatste les komen de leerlingen weer terug bij dezelfde formules die de lessen ervoor zijn behandeld. Deze les heeft een iets andere vorm dan de eerste twee. Hier is het doel heel erg dat de leerlingen zelfstandig of in tweetallen uitzoeken wat het effect is van bepaalde eigenschappen uit het leerproces voor de code. Dit zelfstandige aspect zorgt ervoor dat het geleerde beter onthouden blijft en als het goed is ook beter begrepen wordt (Duch et al. 2001).

5 Dataverzameling

Om de effectiviteit van deze lessen te meten is er een panel van experts gevraagd wat hun mening is over de lessenserie zoals die in appendix B beschreven staat. Dit panel bestond uit vier wiskunde docenten met eerstegraads bevoegdheid. Deze mening is vooral gefocust op de begrijpelijkheid voor de docent van de doelgroep en hoe goed deze lessenserie aansluit op de voorkennis en algemeen niveau van de leerlingen. Dit zijn de onderwerpen waar hier vooral op gefocust zal worden, niet kleine dingen zoals spellingfouten. De antwoorden van de afzonderlijke docenten zijn terug te lezen in appendix A. Naast de opmerkingen van de vier vakdocenten is er ook veel feedback meegenomen van de begeleiders die hebben geholpen in het proces van dit onderzoek.

Positieve opmerkingen

Als eerste bespreek ik de positieve punten die naar voren kwamen van de docenten:

- Unaniem was de opmerking dat het onderwerp een goede manier is om interesse op te wekken bij de leerlingen.
- De docenten zelf waren ook erg geïnteresseerd in het onderwerp.

Verbeterpunten

Hier volgen de verbeterpunten die naar voren kwamen:

- Het gebruik van Engelse termen kan ervoor zorgen dan de leerlingen in de war raken.
- Vooral aangezien de stof, met partiële afgeleiden en sommatietekens, erg hoog gegrepen is voor de leerlingen.
- Tevens zijn de vector berekeningen die hier gebruikt worden een stuk ingewikkelder dan de berekeningen waar de leerlingen bekend mee zijn. Die verwachten meer expliciete vector berekeningen, ten opzichte van de meer algemene berekeningen die nu zijn gegeven.
- Daarbovenop is de docentenhandleiding niet toereikend voor de docenten om de leerlingen fatsoenlijk te begeleiden. Er moet meer achtergrondkennis gegeven worden, zowel over het onderwerp, als het gebruik van de Python bestanden.
- Een laatste opmerking is dat het niet handig is om te beginnen met het onderwerp classificatie, aangezien leerlingen tijdens de wiskunde les niet bekend zijn met dit soort problemen. Regressie daarentegen deelt heel veel eigenschappen met de standaard problemen van het opstellen van een lijn door twee punten.

5.1 Concrete verbeteringen

Hier volgen de concrete verbeterpunten waar de nieuwe versie op is gebaseerd:

- Bijna alle Engelse termen zijn vervangen voor Nederlandse woorden. Hiermee wordt eventuele verdere onderzoek van de leerlingen naar dit onderwerp moeilijker, aangezien daarbij vooral Engelse termen gebruikt worden, maar door de leerlingen de vertaalslag van de Engelse term naar het Nederlands niet te laten maken, zouden de leerlingen deze lessenserie beter aan moeten kunnen laten sluiten bij hun voorkennis.

- Een hele hoop formules zijn versimpeld en explicieter gemaakt. Waar de eerste versie voor de aanpassingen veel algemene formules gebruikt met vector notaties, zijn die in de nieuwe versie opgesplitst in de afzonderlijke coördinaten van dit specifieke probleem. Op dezelfde manier zijn de coördinaten van de datapunten uitgeschreven in x en y (eventueel met w als doelwaarde), ten opzichte van de x_1 en x_2 als coördinaten en y als doelwaarde, zoals gebruikelijker is. Dit, vergelijkbaar met het vorige punt, was gedaan met als doel om de nieuwe stof zo veel mogelijk aan te laten sluiten op de stof waar de leerlingen al bekend mee waren, aangezien de stof zelf waarschijnlijk moeilijk genoeg is om te begrijpen zonder deze aansluiting.

- Veel achtergrondkennis is toegevoegd aan de docentenhandleiding met extra informatie die niet direct in de les wordt beschreven, maar wat wel bij kan dragen aan het begrip van de leerlingen, of wat gebruikt kan worden om de nieuwsgierige leerling wat meer de reden achter bepaalde stappen uit te leggen.

- Sommige van deze achtergrond informatie werd wel belangrijk genoeg geacht om toe te voegen aan de lessenserie zelf, en is daarom hierin verwerkt om het begrip van de leerlingen direct te verbeteren.

References

Bishop, C. M. (2006*a*), *Pattern Recognition and Machine Learning*, Springer, chapter 3.

Bishop, C. M. (2006*b*), *Pattern Recognition and Machine Learning*, Springer, chapter 4.

CLO (2020), 'Zeespiegelstijging langs de nederlandse kust en mondiaal, 1890-2018'.
URL: *https://www.clo.nl/indicatoren/nl0229-zeespiegelstand-nederland-en-mondiaal*

Duch, B. J., Groh, S. E. & Allen, D. E. (2001), 'Why problem based learning? a case study of institutional change in undergraduate education', pp. 3–11.

Freudenthal, H. (1991), *Revisiting Mathematics Education*, Kluwer Academic Publisher.

Harrington, P. (2012), *Machine Learning in Action*, Manning, chapter 5.

Ketkar, N. & Moolayil, J. (2021), *Deep learning with Python*, 2 edn, Apress, chapter 3.

Marzano, R. J. & Miedema, W. (2018), *Leren in vijf dimensies*, 7 edn, Koninklijke van Gorcum B.V.

Rahimi, E., Barendsen, E. & Henze, I. (2018), An instructional model to link designing and conceptual understanding in secondary computer science education, *in* 'Proceedings of the 13th Workshop on Primary and Secondary Computing Education', ACM, pp. 67–70.

Schoenfeld, A. H., Floden, R., "the Algebra Teaching Study & Project", M. A. (2014), 'An introduction to the teaching for robust understanding (tru) framework'.

Skemp, R. (2016), 'Relational understanding and instrumental understanding', *Mathematics Teaching in the Middle School* (2), 88–95.

van Ast, M., de Loor, O. & Spijkerbroek, L. (2021), *Effectief leren*, 5 edn, Noordhoff.

Vygotski, L. (1978), 'Mind in society'.

Wasserman, N. H. (2015), 'Unpacking teachers' moves in the classroom: navigating micro- and macro-levels of mathematical complexity', *Educational Studies in Mathematics* **90**, 75–93.

Part III
Appendices

A Interviews

De dataverzameling vond plaats door een panel van experts, in dit geval eerstegraads wiskundedocenten, via email een vragenlijst te laten beantwoorden. Deze vragenlijst was als volgt:

- Als docent zonder voorkennis over dit onderwerp, in hoeverre legt de lessenserie, zoals die er nu voorstaat, het onderwerp duidelijk genoeg uit dat jij deze zelfverzekerd zou kunnen geven? Wat is duidelijk en wat niet?
- Als docent met ervaring met het lesgeven van de doelgroep (5 vwo wis B leerlingen), in hoeverre sluit, naar jouw mening, deze lessenserie aan op het niveau van de leerlingen? Welke dingen passen wel en wat is te moeilijk?
- Los van de aansluiting op het niveau van de leerlingen, hoe duidelijk wordt de stof uitgelegd aan de leerlingen en wat zou beter kunnen?
- Het doel van de lessenserie is om het nut van wiskunde in de hedendaagse samenleving duidelijk te maken en leerlingen wat meer geïnteresseerd krijgen in wiskunde. In hoeverre is dat hiermee geslaagd?
- Zijn er nog andere adviezen/opmerkingen die je kwijt wil?

Veel van de docenten hebben ervoor gekozen om de vragenlijst meer in een verhalende vorm te beantwoorden of de lessenserie in volgorde te beoordelen in plaats van de vragen per stuk te beantwoorden. Ondanks dat worden de vragen wel beantwoord in de e-mails. Daarom is er alsnog voor gekozen om de data zo te gebruiken.

Docent 1

Opmerkingen lessenserie Ruben Snijders AI

Algemene feedback Ruben, je hebt een heel leuk onderwerp gekozen om een lessenserie over te maken. Het is een onderwerp dat veel leerlingen zeker zal boeien. Je hebt ook je best gedaan om de lessenserie divers te maken: veel verschillende onderwerpen passeren de revue. Daarnaast is het tof dat je ook een Python script hebt geschreven zodat leerlingen zelf kunnen experimenteren. Er zijn wel enkele belangrijke verbeterpunten. Zoals de lessenserie er nu ligt, zou ik deze niet geven. Een belangrijk aandachtspunt is de voorkennis. Zo kun je van bijv. docenten en leerlingen niet maar zo veronderstellen dat ze Python kunnen. Ook partiele afgeleiden kennen de leerlingen niet (sommaties?). Een ander aandachtspunt is dat er nu weliswaar veel onderwerpen worden behandeld, maar de uitleg kan zeker nog secuurder en volledi ger. Mijn algemene advies is daarom: Less is more. Ik zou minder behandelen, maar met rustiger opbouw, en uitgebreide aandacht voor alle nieuwe begrippen en notatie. Zie hieronder uiteenlopende opmerkingen om het geheel te verbeteren. Het lijkt mij heel tof als er uiteindelijk een mooie, volledige en zorgvuldig opgebouwde lessenserie over AI uitkomt.

I. LEERLINGENBLAD

Les 1

- 'Classificatieprobleem'. Geef direct een voorbeeld van een 'klasse' en wat de 'classificatie' hier inhoudt. De algemene regel: bij elke technische term een voorbeeld.
- Figuur 1. Hoe ga je om met het verschil de eenheden voor de x-as en y-as? In hoeverre is dat hier van belang? (en wat is de bron van data?).
- Graag meer uitleg over we scheidslijn. Wat doe die precies? Wat betekent het om aan een kant van de lijn te liggen? Geef evt. verschillende scheidslijnen en vraag de lln te kiezen welke de beste is.
- 'Deze lijn is weergegeven door w0.' Zien we die ergens? Graag ergens in een diagram tekenen.
- Suggestie: geeft een tabel met data en laat zien wat je met x_1 bedoelt. Maak inzichtelijk dat dit inderdaad een vector is met allerlei features (zoals lengte, gewicht etc..).
- 'In ons geval wordt dit gedaan door de vector van de lijn (w0) te vermenigvuldigen met de vector van de eigenschappen van het eerste datapunt (x1).' Kun je uitleggen waarom dit zo werkt? Wat is de geometrische intuïtie hierbij? Wat bedoel je precies met de 'vector van de lijn'? Een leerling denkt aan richtingsvector en steunvector. Dat zijn twee vectoren bij 1 lijn?!
- 'Daarnaast is w = (w0, w1, w2), waar w1 en w2 de verhouding tussen het gewicht en de lengte is van de scheidingslijn.' Dit is onduidelijk. Vogels

hebben gewicht en lengte, maar de scheidingslijn toch niet. Wat duiden w1 en w2 precies aan?

- 'weight vector'. Even uitleggen wat dit inhoudt. Begrip is nieuw voor ons. Dit begrip komt nu uit de lucht vallen. Dit stukje herschrijven.
- 'per definitie'. Weglaten.
- Tekstueel: zinnen wat herformuleren, zodat ze korter en eenvoudiger zijn. Checken of de zinnen goed lopen. (Bijv. "Het is zelfs onmogelijk om een rechte lijn te trekken die beide groepen perfect scheiden."). Alles controleren op d/t-fouten.

Les 2

- Suggestie: begin met lineaire regressie.
- Goed idee om de leerlingen de lijn zelf te laten trekken!
- Suggestie bij je benadering met polynomen: laat leerlingen zelf bedenken wat de beste benadering/lijn is voor M=0 (evenwichtsstand).
- 'Error'. Probeer alles zoveel mogelijk in het Nederlands te zeggen. Engels is voor sommige leerlingen een extra belasting. Dat gaat ten koste van de focus op wiskunde.
- Formule 2. Je sommatie begint bij i=0 (dat lijkt me ongebruikelijk). Klopt het dat je N+1 datapunten hebt?
- Formule 3. Leg gradient descent helemaal uit:
  - Wat representeert de afgeleide precies?
  - Wat is eeta?
  - Vanwaar het minteken?
  - Dat doet de update dus in zijn geheel?
- Formule 3. Let op: als je dit goed wilt uitleggen, heb je partiele afgeleiden nodig, die kennen de leerlingen niet. Vraag: kennen ze wel het sommatieteken?
- "Zelfs als we een computer de beste lijn laten vinden, is dit het beste geval." Wat bedoel je hier?
- 'logische lijn'. Ik weet niet wat dit is.
- Opdracht 2 liever weglaten. Eerder focussen op een goede uitleg van opdracht 1.

II. DOCENTENHANDLEIDING

- Graag meteen aan het begin zeggen wat de doelgroep is. Dat is: 5 Wis B en hoger.
- Vraag: welke voorkennis verwacht je bij de docenten? Bijvoorbeeld:
  - Verwacht je dat elke Wiskunde-B-docent bekend is met regressie? (Ik denk niet dat iedereen dat kent; ook al is de techniek eenvoudig; regressie zit soms in het WisD programma)
  - Verwacht je dat docenten op de hoogte zijn van de wiskunde achter een neural nets (NN) (d.w.z.: input vs. bias, gewichten, activatiefunctie etc.)? Ik zou dit ofwel helemaal uitleggen, ofwel weglaten. Dat laatste is waarschijnlijk het beste.
  - Verwacht je dat de docent kan programmeren in Python (de meeste docenten zullen daar niet mee bekend zijn). Als een docent dat niet kan, welke instructies/begeleiding geef je hem/haar dan?
- Het stuk over NNs komt op een vreemde plaats in de tekst: docenten weten nog niet wat de werkvorm precies behelst, maar moeten zich nu al verdiepen in een uitbreiding/verbreding. Liever eerst de werkvorm volledig uitleggen.
- Afbeelding NN:
  - Waarom zijn alleen sommige gewichten genummerd?
  - Welke gedachte zit er achter de volgorde van de gewichten?
  - Doe je geen bias? Waarom niet?
- Tekstueel: tekst nalezen op typos ('Elke leerling berekend...'). Sommige zinnen lopen niet lekker (bijv: '' Persoonlijk vind ik een woordwolk met bijvoorbeeld Wooclap een goede manier om dit te doen aangezien dit ook weergeeft als een antwoord vaak gegeven wordt.'). Woorden aan elkaar schrijven i.p.v. los op z'n Engels ('regressie probleem'), etc..

III. Coderen

- Gaan de lln in les 1 al coderen? Dat lijkt de titel van 2.3 te suggereren. Hieronder krijgen we een beschrijving van de 3 lessen. Titel lijkt de inhoud dus niet te dekken.

Les 1

- Het zou handig zijn om een schema/lesplan te hebben van de 3 lessen en de indeling van die lessen. Het overzicht van de tijdsverdeling komt nu later, maar ik denk dat je er beter mee kan beginnen.
- In het algemeen: ik denk dat je je uitleg van AI kunt verbeteren door steeds een conceptueel onderscheid te maken tussen trainingsset en testset.

- Jouw uitleg over classificatie etc. zou ik een apart kader in de handleiding doen. Dan is de tekst mooi verdeeld tussen instructies aan de docent vs. instructie/onderwijs aan de leerlingen. Het is dan voor de docent ook makkelijker om op te zoeken wat hij gaat vertellen in de les wanneer hij de docentenhandleiding openslaat.
- Herhaal in de docentenhandleiding ook kort waar de dataset voor staat etc..
- 'de nieuwe weight vector berekend en bekeken worden'. Hoe gaat dat dan precies? Weet de docent hier al wat een weight vector is en waarom je die zou gebruiken?
- Algemener: ik zou in de docentenhandeling een kort stukje opnemen waarin je uitlegt welk algoritme je hier gebruikt, onder welke naam dat algoritme bekent is, met verwijzingen naar meer achtergrond (Wikipedia is meestal goed). Zorg dat docenten altijd meer informatie kunnen opzoeken als ze aan jouw handleiding niet genoeg zouden hebben.
- 'formule 1' – Wat is formule 1? Waar vind ik die? (Herhaal formule of verwijs naar leerlingblad).
- Wat doet de docent na 1 iteratie? Laat zij er meer zien a.h.v. de code?
- Wordt er na 1 iteratie al iets gezegd over de fouten die je scheidslijn maakt? Wordt de gemiddelde fout kleiner door het verschuiven van de scheidslijn? Zo ja, hoe zien we dat?

Les 2

- Suggestie: noteer aan het begin van deze handleiding per les wat de lesdoelen zijn. Die kunnen inhouden dat de leerlingen bepaalde begrippen leren. Dat lijkt in jouw les 2 ook zo te zijn; tot die begrippen behoren overfitting, underfitting, (de trade-off ertussen).
- Suggestie: herhaal wat het doel is van AI of statistical learning. Dat is: een functie leren op basis van datapunten. Benadrukken dat de data bekend zijn, maar dat de functie onbekend is. Plaatjes zoals Figuur 14 kunnen dat heel goed in beeld brengen! Toevoegen: we spelen even dat de functie onbekend is en genereren die met een sinus en ruis (wat voor ruis trouwens?). Leerlingen moeten eerst begrijpen dat dit het hele doel van statistical learning is.
- Suggestie: laat in de docentenhandleiding eerst de uitwerkingen zien. Dus: laat zien wat de docenten (samen met de leerlingen) uiteindelijk moeten krijgen. Daarna kun je naar die resultaten verwijzen in je tekst.
- 'De eerste' – eerste wat?. Beter: afbeeldingen toevoegen, daarnaar verwijzen. Nu hebben we de uitwerkingen niet voor de geest.

- Welke data? Verwijs naar de je datasets, vermeld wat ze inhouden.
- "Als er tijd is kan er een discussie geopend worden over het belang van de vorm van functie 2 en het gebruik van de afgeleide hiervan." Ik denk dat het van belang ik om je werkvorm zo makkelijk mogelijk te maken voor de uitvoerende docent. Hier zou je wat specifieke parameterwaarden kunnen toevoegen waarmee de docent de effecten kan laten zien die jij beschrijft.
- 'In dezelfde lijn kan'. Nieuwe complicatie, dus nieuwe alinea in de tekst.
- Suggesties bij Figuren 13 en 14. Schrijf in de caption daarbij wat we hier zien? Wat zijn de parameterwaarden? Welk fenomeen illustreer je met de figuren? Bij Figuur 14 nauwkeuriger zijn: alleen bij M=9 hebben we te maken met duidelijke overfitting.
- Ook hier: zinnen korter en bondiger maken. Sommige zinnen lopen niet ("De eerste [regressieprobleem] heeft al een lijn erdoor, de tweede is een voorbeeld waar data gegenereerd is van een sinus en van de leerlingen gevraagd wordt om een lijn hier doorheen te trekken." Of: "Echter, als het model te ingewikkeld wordt gemaakt, kan het ook alle representatie van de werkelijkheid (uitgebeeld in figuur 14)"). Dit soort niet-lopende zinnen zijn passim.
- Typesetting: De indices van gewichten etc. staat niet overal netjes.

Les 3 (Programmeren)

- Lesdoel is wat vaag. Vgl: 'Het doel van de les is dat leerlingen het belang van bepaalde stappen gaan onderzoeken'. Welke stappen? Welk belang? Wordt erna duidelijker, maar liever bondig formuleren.
- Wederom: de zinnen zijn niet-grammaticaal ('De vindingen die ik verwacht dat de leerlingen zullen vinden is dat met een kleinere waardes in de begin weight vector, er sneller een grote error uit komt en dat eta kleiner moet worden gemaakt.'
- 'Overshoot'. Is dat een term die we al gezien hebben? Kennen docenten deze term? Graag alle technische begrippen uitleggen.

Algemene opmerkingen bij docentenhandleiding

- Docenten die nog niet zo thuis zijn in de AI of statistical learning, zullen zeker behoefte hebben aan meer wiskundige achtergrond. Geef achtergrondinformatie, verwijs naar Wikipedia, gebruik geen technische termen zonder ze te introduceren.
- Docenten zullen profiteren van heldere en volledige uitwerkingen van de opdrachten.

- Het is nuttig om evt. discussie te stimuleren door aan de docent alvast specifieke relevante parameterwaarden mee te geven waarmee ze interessante situaties kunnen laten zien. Maak het de docent zo makkelijk mogelijk! Lever evt. een ppt aan met relevante figuren zodat de docent deze direct kan laten zien (of bestand met figuren).

- Bedenk nog eens wat de lesdoelen bij elke les nu precies zijn. Daar zitten heel wat begrippen in die nu nog niet expliciet als lesdoel genoteerd staan. Dat geeft zicht op hoeveel je in een les wilt doen en dus op welke les mogelijk te vol zit. M.i. zit les 2 te vol.

III. VERANTWOORDING

- Black box houdt niet in dat programmeurs hun code niet begrijpen (dat doen ze wel), maar dat er geen eenvoudig interpretatie of vertaalslag te maken is voor de bewerkingen die het algoritme uitvoert naar de wereld van verschijnselen. Programmeurs snappen wat de gewichten in een NN doen, maar hoe die gewichten precies de relevante structuren in de data representeren – dat is onbekend.

- Wat bedoel je precies met het verschil tussen 'programmeren' en 'blindelings programmeren'? Vraag: weten leerlingen hoe bij de lineaire regressie de 'beste' regressielijn wordt gevonden? Er wordt verwezen naar een lijn die de computer vindt. Maar weten leerlingen hoe de computer daaraan komt?

- De verantwoording kan wel wat concreter.

IV. PYTHON SCRIPT

- Vraag: weten leerlingen hoe ze met Python moeten werken of hoe ze in Notebooks moeten werken? Snappen ze hoe ze commando's runnen etc..? Zo niet, dan moet er minstens een begeleidende tekst bij (en wat voorbeelden) of hoe je tekstblokken runt.

- Vraag: welk stuk moeten de leerlingen lezen/begrijpen? In hoeverre wil je dat de leerlingen de code kunnen begrijpen?

- Vraag: Welk stuk moeten de leerlingen runnen?

- Vraag: zitten leerlingen hiervoor in een computerlokaal? Of nemen ze zelf computers mee? Zo ja, is Anaconda o.i.d. daar dan geïnstalleerd? Graag opmerkingen hierover in de docentenhandleiding.

Docent 2

Hallo Ruben,

Wat een interessant onderwerp en wat leuk dat ze een keer wat anders zien dat echt met wiskunde te maken heeft, heel actueel én de toekomst is.

Ik denk echt dat je de les een keer moet geven om het helemaal te optimaliseren, en dat er ook wel de nodige uitleg van de docent nodig is. Hieronder een aantal aandachtspunten:

Geschikt voor eind 5v. in getal en ruimte is H10 vectoren nodig en dit komt in februari ongeveer aan bod. Kan evt als keuzeonderwerp dat altijd gegeven moet worden als SE stof in 5v wi B. dan moet je dit ook toetsen als SE toets.

- Blz. 1 begrip puntenwolk kennen ze waarschijnlijk niet (of weten ze niet meer), maar met toelichting docent moet dat lukken uiteraard.
- Blz. 3 Sommatie-teken, kennen ze waarschijnlijk niet. Dit soort formules vinden ze lastig om te lezen.
- Blz. 3 komt uit de lucht vallen, maar ik zag verderop de lijst met definities. Misschien naar verwijzen of vooraan zetten?
- Blz. 3 e.a. je gebruikt overal veel 'moeilijke' woorden. Overschat de woordenschat van een 5v lln niet. Woorden als arbitrair, numeriek, weight vector, update, iteraties. In mijn eigen 5v kende een aantal lln het het woord 'interpreteren' niet eens...
- Blz. 3 tabel 1.1.1 Welke is dit?
- Blz. 3 formule 4 Welke is dit? Hierdoor is mij de opdracht niet duidelijk.
- Blz. 11 zelfde, formule 4? functie 1?
- Blz. 11 en blz 15. bekijk de veelgebruikte wiskunde methodes(getal en ruimte/ moderne wiskunde) en noteer formules zoals de middelbare school boeken het doen, met dezelfde symbolen ed. Ik ben bang dat lln anders afhaken. Noteer het woord kettingregel bij blz. 15 erbij. Dit is te wiskundig genoteerd. Vectoren noteren we ook anders op de middelbare school. En afgeleide

  Dit geldt ook voor de lay-out/lettertype. Mag wel iets aansprekender((vind ik persoonlijk), bekijk ook eens de wiskunde methodes. Maar hierover kun je discussiëren, lln voorbereiden op de uni dan is dit misschien ook wel een keer goed.
- Blz. 16 voorspellingslijnen, let op het afkorten
- Blz. 17 voor NOTE mist tekst
- Blz. 19 typefout can ipv van
- Blz 19 mooi voorbeeld van de schilders om dan te gebruiken in de les voor de docent

Ik denk wel dat je als docent goed in deze materie moet zitten om het helemaal uit te kunnen voeren.

Als ik je de pdf bestanden van de getal en ruimte boeken moet sturen, laat maar weten. Dan kun je zien hoe ze het gewend zijn en wat de voorkennis is.

Hopelijk kun je er wat mee, veel succes ermee!

Docent 3

Hey Ruben,

Erg leuk idee. Een aantal opvallende punten in een willekeurige volgorde.

De cheat sheet is een mooi idee. Ik denk alleen dat het eerder een uitleg van definities is, die nu ontbreekt in de lessen op de werkbladen. Wiskunde B vwo-leerlingen zullen op de middelbare school een behoorlijke kluif hebben aan alleen al het begrijpen wat hier staat. Zij hebben geen lineaire algebra. Kijk hier maar eens. Dit is de examenstof voor meetkunde met vectoren en dergelijke. http://wm.math4allview.appspot.com/view?comp=lj6-v-b-h16&subcomp=lj6-v-b-h16-08&variant=basis_wm&item=1. Dit is veel concreter dan bijvoorbeeld: a+b=(a0 + b0, a1 +b1, ..., a(n)+b(n)) Deze formele manier van notatie komen ze op de middelbare school niet echt tegen.

De tekst op pagina 3 van " Dit wordt gedaan met de volgende formule..... tm opdracht 1" is echt veel te abstract voor onze leerlingen. Dit begint al met het sommatie teken wat ze een keertje gezien hebben bij riemansommen maar daarna nooit meer mee gerekend hebben.

Ik snap de opbouw van de lessen, maar is het niet mogelijk om toch met regressie te beginnen. Dit sluit het meest aan op de belevingswereld en is het minst abstract naar mijn idee.

Bij figuur 8 laten zien dat M staat voor de hoogste macht.

2.4 programmeren zou ik ook les 3 erbij noemen

Kortom vind ik het nog steeds een erg leuk idee, maar zoals het nu in elkaar zit sluit het nog niet aan op het niveau van middelbare scholieren. Je zult echt moeten kijken waar je misschien nog meer kan downgraden. Volgens mij heb je dat voor je gevoel dat al wel wat gedaan, maar het moet nog meer. Probeer het over het algemeen wat meer te concretiseren. Het einddoel kan wellicht ook behaald worden met minder abstracte wiskunde. Als je dit wel laat staan ben je klassikaal echt wel veel tijd kwijt met abstracte wiskunde begrijpelijk te maken. Als is het alleen al om goed uit te leggen hoe formule 2 in elkaar steekt op bladzijde 11.

Hoe nu verder: als 5A wis B je publiek blijft dan zal het een stuk concreter moeten ben ik bang. Ook al geef je nog wel de instructies. Denk na of er bepaalde dingen klassikaal uitgelegd kunnen worden zonder al te diep de wiskunde in te duiken. Je publiek veranderen. Je snapt misschien dat ik dit liever mondeling met je had besproken, dan heb jij ook gelijk de kans om te reageren. Nu komt het misschien wat bot over. Dus dat kan altijd nog.

*Docent 4 Hey Ruben,

Wat een leuk idee als onderwerp voor een lessenserie!

Hoewel de tekst zoals je die aan de leerlingen wil geven wel duidelijk is, denk ik dat, om dit goed aan leerlingen uit te kunnen leggen, het wel nodig is om extra achtergrondkennis te hebben. Zo kan ik me voorstellen dat leerlingen veel vragen gaan stellen waar ik niet direct antwoord op zou kunnen geven. De docenten handleiding helpt hier wel mee, maar niet genoeg om alles te beantwoorden.

Ik denk dat leerlingen wel veel moeite gaan hebben met bepaalde onderwerpen die je hierin introduceert. Hierbij kijk ik voornamelijk naar de partiële afgeleiden en de vectoren die je niet helemaal uitschrijft met lengte n. Daarnaast sluit de manier waarop je het hebt ontworpen niet helemaal aan met de lesmethodes die de leerlingen gewend zijn zoals Getal en Ruimte of Moderne Wiskunde. Het kan geen kwaad om ze kennis te laten maken met zo een universitair ontwerp, maar daarmee sluit het minder aan op wat zij gewend zijn. De voorbeelden die je gebruikt zijn wel aansprekend voor de leerlingen.

Het is een hele leuke manier om de leerlingen een voorbeeld te geven van hoe wiskunde gebruikt kan worden vandaag de dag, maar het moet nog wel echt vereenvoudigd worden om goed aan te sluiten op de doelgroep

B Leerling werkbladen

B.1 Les 1: Classificatie

Deze komende twee lessen zullen we behandelen hoe wiskunde die jullie geleerd hebben in de lessen gebruikt kan worden om computers iets te laten leren. Hier zullen jullie als leerlingen eerst de rol spelen van een computer en de stappen doorlopen om een voorspelling te maken. ¶¶ Daarna wordt uitgelegd hoe een AI (Artificial Intelligence, oftwel Kunstmatige Intelligentie) dit automatisch kunnen doen en op veel ingewikkeldere problemen.

Deze les focust op een bepaald soort probleem, classificatie. Classificatie problemen zijn problemen waar een voorspelling wordt gedaan of iets tot een bepaalde groep behoort of niet. Dit is terug te zien aan de term "class" in classificatie, wat groep betekent. Het doel van AI in een classificatie probleem is het scheiden van groepen met behulp van lijnen.

Voordat we gaan kijken hoe een AI zulke lijnen trekt, gaan we eerst zelf eens proberen groepen op deze manier te onderscheiden. Neem de grafiek met de lengte van de werkers en de lengte van de koningin van bijen en wespen. Trek nu een rechte lijn om deze twee van elkaar te scheiden.

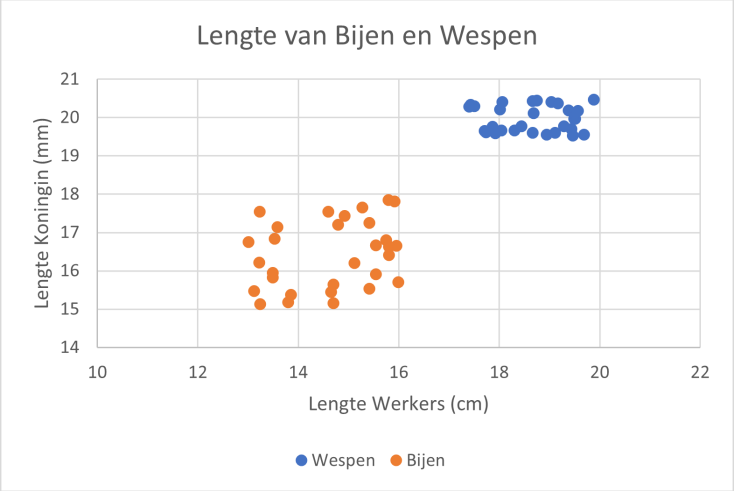

Figure 16: Puntenwolk van korven wespen en korven bijen

¶¶Deze manier kan bijvoorbeeld gebruikt worden door Netflix, waar gebruikers ingedeeld kunnen worden in groepen gebaseerd op films die ze kijken, locatie en andere eigenschappen, om films aan te raden op een iets slimmere manier dan alleen dezelfde genres.

Zoals je ziet is dit niet al te moeilijk. Het is zelfs mogelijk om heel veel verschillende rechte lijnen te trekken die nog steeds alle punten goed verdelen.*** Echter is het niet altijd zo makkelijk. Probeer nu hetzelfde te doen voor het voorbeeld met het gewicht en de lengte van spechten en roeken.

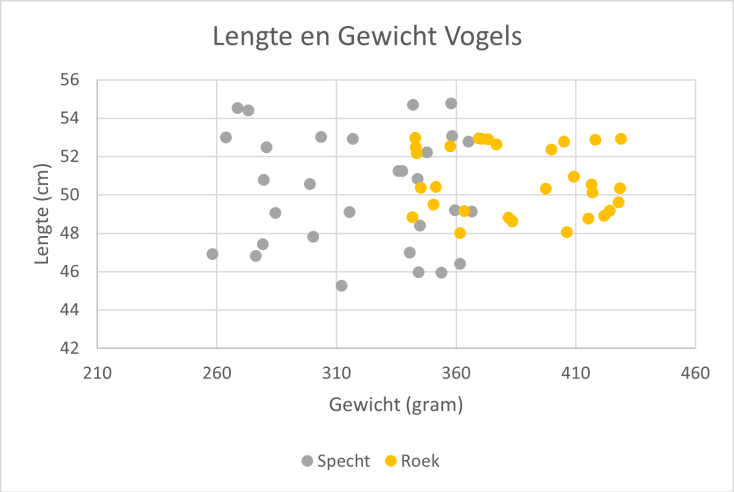

Figure 17: Puntenwolk van de lengte en het gewicht van spechten en roeken

Dit is een stuk moeilijker. Het is zelfs onmogelijk om een rechte lijn te trekken die beide groepen perfect scheiden. In dit geval is het doel om zo min mogelijk punten aan de verkeerde kant van de lijn te hebben.

†††Zoals je ziet is het handmatig wel goed te doen om een lijn te trekken tussen twee groepen die gedefinieerd zijn door twee eigenschappen. Echter zijn heel veel dingen niet op deze manier te classificeren. Sommige dingen zijn bijvoorbeeld afhankelijk van meer eigenschappen om goed te kunnen visualiseren, waardoor het niet zo makkelijk handmatig te classificeren is. Hierom is het gebruik van AI om goede classificaties te maken ontwikkeld.

Laten we nu kijken hoe een AI automatisch een goede lijn kan trekken. De AI begint met een gok voor deze lijn. Deze lijn is weergegeven door $\mathbf{w}_0$. Vervolgens kijkt de AI naar alle datapunten en maakt een voorspelling over deze datapunten. In ons geval wordt dit gedaan door de vector van de lijn ($\mathbf{w}_0$) te vermenigvuldigen met de vector van de eigenschappen van het eerste datapunt ($\mathbf{x}_1$). Als de uitkomst positief is wordt voorspeld dat het datapunt tot de ene groep behoord en als de uitkomst negatief is tot de andere. Deze

***Er zijn manieren om de beste lijn te bepalen in zo een geval. Hiervoor wordt de afstand tussen de lijn en het dichtstbijzijnde punt van elke groep gemaximaliseerd.

†††Soms weten we niet eens wat precies de categorieën zijn of waar we naar zoeken. Zo kan bijvoorbeeld informatie gevonden worden door data te verdelen in groepen die zo dicht mogelijk bij elkaar ligt, zonder echt te weten wat de gemeenschappelijke eigenschap is, als die er al is. Dit is unsupervised learning, in tegenstelling tot supervised learning wat we deze lessen behandelen.

uitkomst wordt vergeleken met de werkelijke waarde die bij dit datapunt hoort (t_1) en, als dit niet overeenkomt, beweegt de AI de lijn in de richting van de foute datapunten, zodat die dichter bij de andere, juiste, kant komen te liggen.

Dit wordt gedaan met de volgende formule:

$$\mathbf{w} = \mathbf{w}_0 + \eta \sum_{i=0}^{N} (\mathbf{x}_i \cdot t_i) \tag{9}$$

In het voorbeeld hierboven met de vogels, datapunt 2 is bijvoorbeeld een vogel een gewicht van 410 gram en een lengte heeft van 51 cm, dan is $\mathbf{x}_2 = (1, 410, 51)$. Daarnaast is $\mathbf{w} = (w_0, w_1, w_2)$, waar w_1 en w_2 de verhouding tussen het gewicht en de lengte is van de scheidingslijn. Dit bepaalt hoe steil de scheidingslijn komt te liggen. w_0 is een verschuiving naar boven of beneden. Vandaar dat het bijbehorende coördinaat in $\mathbf{x}$ altijd een 1 is, omdat deze verschuiving onafhankelijk van $\mathbf{x}$ is.

Hier wordt de weight vector dus aangepast als $\mathbf{w}_i \cdot \mathbf{x}_i$ niet een uitkomst geeft die overeenkomt met t_i. Een AI kan dit makkelijk doen door te controleren of $t_i \mathbf{w}_i \mathbf{x_i} < 0$. Als $\mathbf{w}_i \cdot \mathbf{x}_i$ niet overeenkomt met t_i dan is de ene positief en de ander negatief, waardoor het product van die twee per definitie negatief is als een punt verkeerd ingedeeld wordt. N is hier het aantal verkeerd ingedeelde datapunten in de dataset. Deze aanpassing, het $\eta \sum_{i=0}^{N} (\mathbf{x}_i \cdot t_i)$ gedeelte, komt erop neer dat de vector van eigenschappen, oftewel een deel van de coördinaten van het verkeerd geclassificeerde datapunt ($\mathbf{x}_i$), wordt opgeteld of afgetrokken (bepaald door t_i) van de weight vector van de lijn. Hoe een groot deel wordt bepaald door η. Merk op dat de uitkomst van $\eta \sum_{i=0}^{N} (\mathbf{x}_i \cdot t_i)$ een vector is.

Om dit iets duidelijker te maken volgen jullie nu een praktisch voorbeeld:

B.1.1 Opdracht 1

Of leerlingen een voldoende gaan halen op een toets in een toetsweek is afhankelijk van een hele hoop factoren. Dit zijn onder andere hoeveel tijd een leerling heeft geleerd, hoe goed een leerling is in een vak, hoe druk een leerling is. Nu zijn de laatste twee hiervan niet makkelijk numeriek weer te geven, maar dit kan gedaan worden door te kijken naar gevolgen, zoals het vorige cijfer en het gemiddelde cijfer voor hoe goed een leerling in een vak is en het aantal toetsen die een leerling die toetsweek heeft voor hoe druk hij is. In de volgende dataset (tabel **??**) is van verschillende leerlingen aangegeven hoeveel toetsen zij die week hadden op de x_1-as en hoe lang zij hebben geleerd in uren op de x_2-as. Deze zijn geven zowel in tabel vorm als in grafiek vorm.[‡‡‡]

De weight vector aan het begin is $\mathbf{w}_0 = (20, 15, -40)$. Dit komt dus overeen met de lijn $20x + 15y = -40$ Dit is erg arbitrair, maar werkt goed met een update voor deze les. In les 3 kunnen jullie experimenteren met andere vectoren als begin vector. η voor dit probleem is $0,7$. Gebruik formule **??** om de

[‡‡‡]Merk op dat, ondanks het feit dat de waardes bij x_1 en x_2 numeriek zijn, en we voldoende aangeven als een 1 en een onvoldoende aangeven als een -1, het nog steeds een classificatie probleem is.

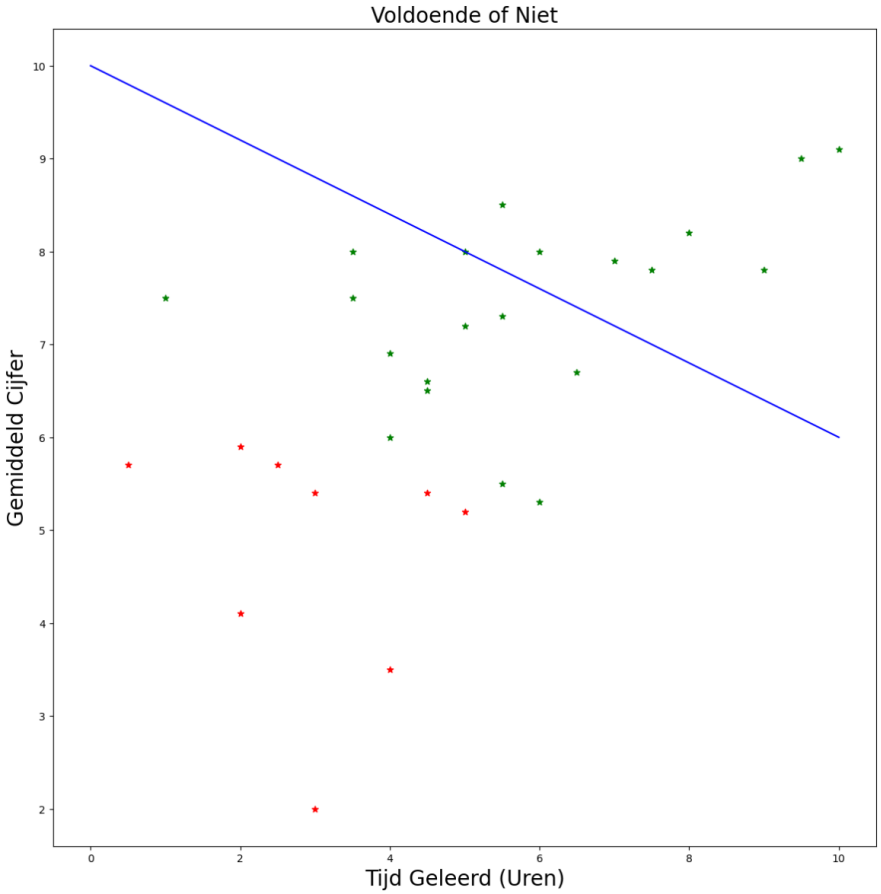

Figure 18: Grafiek voor het classificatie probleem

output te berekenen van elk van deze datapunten. Als er een positieve waarde uitkomt wordt het datapunt geclassificeerd als een +1, oftewel voldoende, en voor een negatieve waarde een -1, oftewel een onvoldoende. Merk op dat er nu gecontroleerd kan worden of een datapunt wel of niet goed geclassificeerd is berekend kan worden door $\mathbf{w}_i\mathbf{x}_i t_i$ te berekenen. Als dit groter is dan 0 wordt het datapunt goed geclassificeerd en vice versa. Voor jullie is dit niet direct nodig, aangezien je het punt in de grafiek op zou kunnen zoeken, maar voor een AI is het wel makkelijk om zo te kunnen berekenen welke er goed zijn en welke niet.

Als alle waardes zijn berekend, update de weight vector aan de hand van formule **??**, met $\eta = 0,7$ en kijk hoe dit de lijn veranderd.

	Tijd Geleerd	Aantal Toetsen	t
0	0.5	1	1
1	1	1	1
2	2	3	-1
3	2	2	-1
4	2.5	4	1
5	3	5	-1
6	3	3	1
7	3.5	2	1
8	3.5	1	1
9	4	3	1
10	4	2	1
11	4	4	-1
12	4.5	3	-1
13	4.5	1	1
14	4.5	2	1
15	5	4	-1
16	5	3	1
17	5	1	1
18	5.5	3	1
19	5.5	2	1
20	5.5	4	-1
21	6	5	-1
22	6	2	1
23	6.5	4	1
24	7	3	1
25	7.5	2	1
26	8	1	1
27	9	5	1
28	9.5	4	1
29	10	1	1

Table 3: Tabel voor het classificatie probleem

B.2 Les 2: Regressie

De vorige les werden classificatie problemen behandeld. Dat waren problemen waar er voorspeld moet worden of een datapunt tot een bepaalde groep behoort of niet. Deze les focust zich op regressie problemen. Dit zijn problemen waar het doel is om een specifieke numerieke waarde te berekenen. Dit wordt gedaan door wederom een lijn te trekken gebaseerd op de data die de AI heeft, maar nu in plaats van de datapunten in tweeën te splitsen is nu het doel de lijn zo dicht mogelijk bij alle data te krijgen. Een goed voorbeeld hiervan is bijvoorbeeld de voorspelling van hoe de zeespiegel gaat stijgen in de komende jaren.

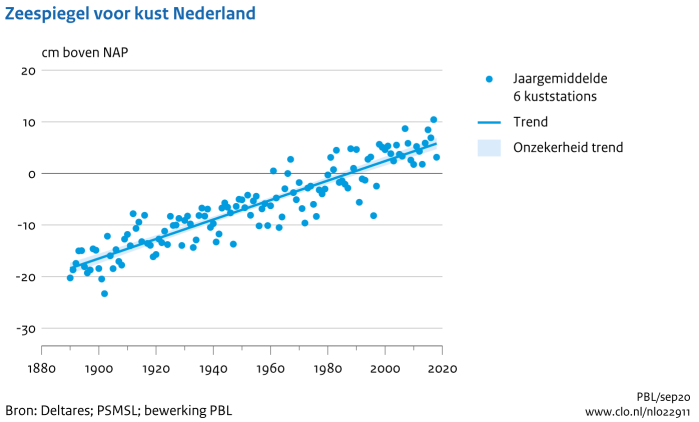

Figure 19: Niveau zeespiegel (CLO 2020)

Hier zie je dat er ongeveer elk half jaar en meting is gedaan van de zeespiegel. Door een lijn te trekken die zo goed mogelijk overeenkomt met de gemeten data, kan deze lijn doorgetrokken worden naar de toekomst om zo voorspellingen te doen over deze toekomstige data. Zoals je ziet gaat de lijn niet door alle punten. Het is vrijwel onmogelijk om zulke dingen exact te voorspellen aangezien er altijd factoren zijn die je niet meeneemt in de meting, al dan niet omdat je ze over het hoofd ziet, of omdat het je model te ingewikkeld maakt. Daarom is het doel om de error zo klein mogelijk te maken met het model dat je hebt.

Neem bijvoorbeeld de volgende dataset (figuur **??**). Deze hypothetische negen datapunten zijn gegenereerd met een specifieke functie en een kleine willekeurige error daarbovenop. Probeer een rechte lijn hier doorheen te trekken die zo goed mogelijk overeenkomt met deze punten.

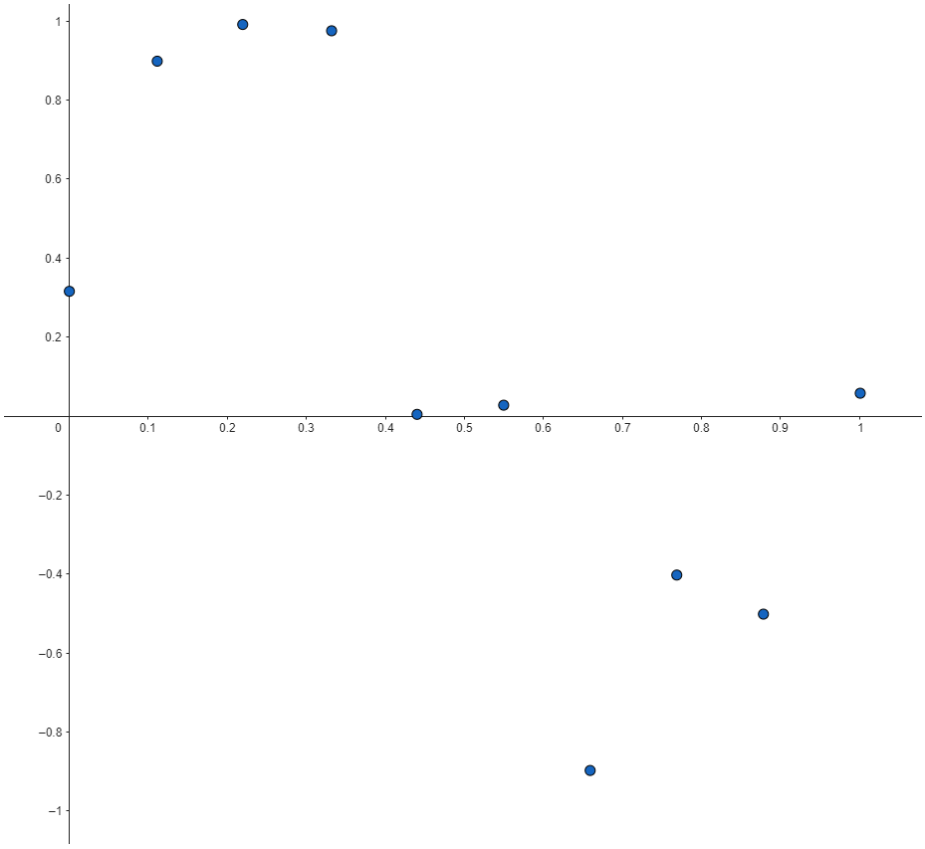

Figure 20: Een hypothetische dataset

Zoals je ziet past een rechte lijn niet heel goed op deze dataset. Zelfs als we een computer de beste lijn laten vinden, is dit het beste geval (figuur **??**).

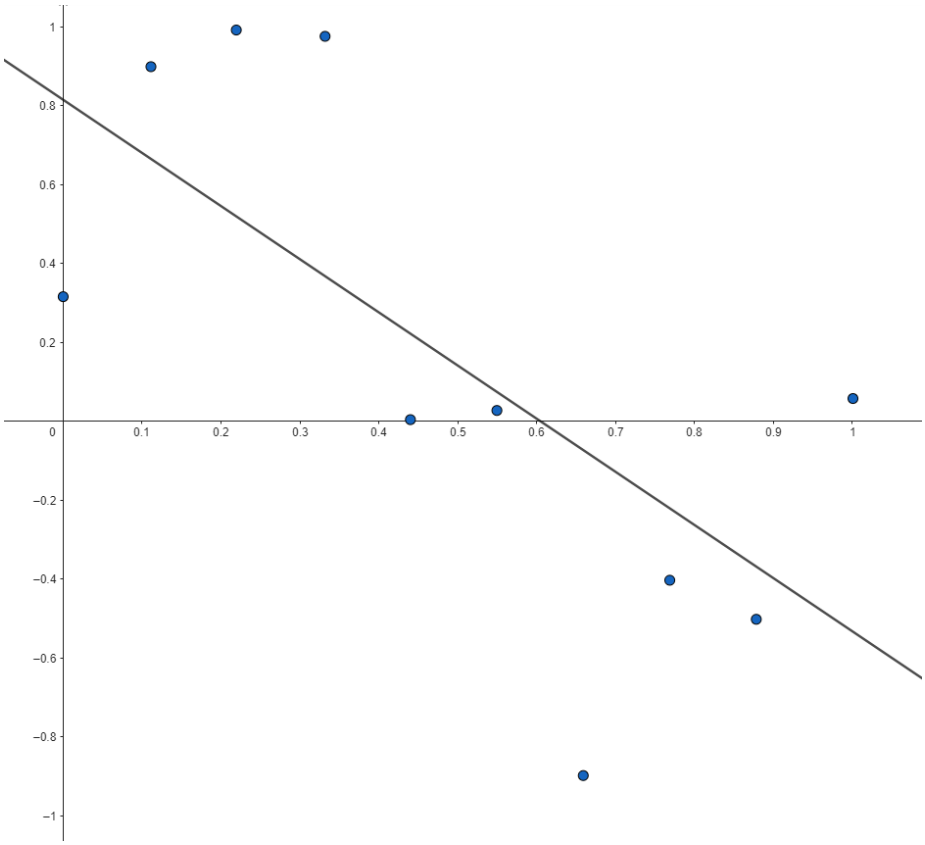

Figure 21: De beste rechte lijn die te trekken valt door deze dataset

De functie waar deze datapunten van afstammen is een sinus met een amplitude en periode van 1 (figuur **??**).

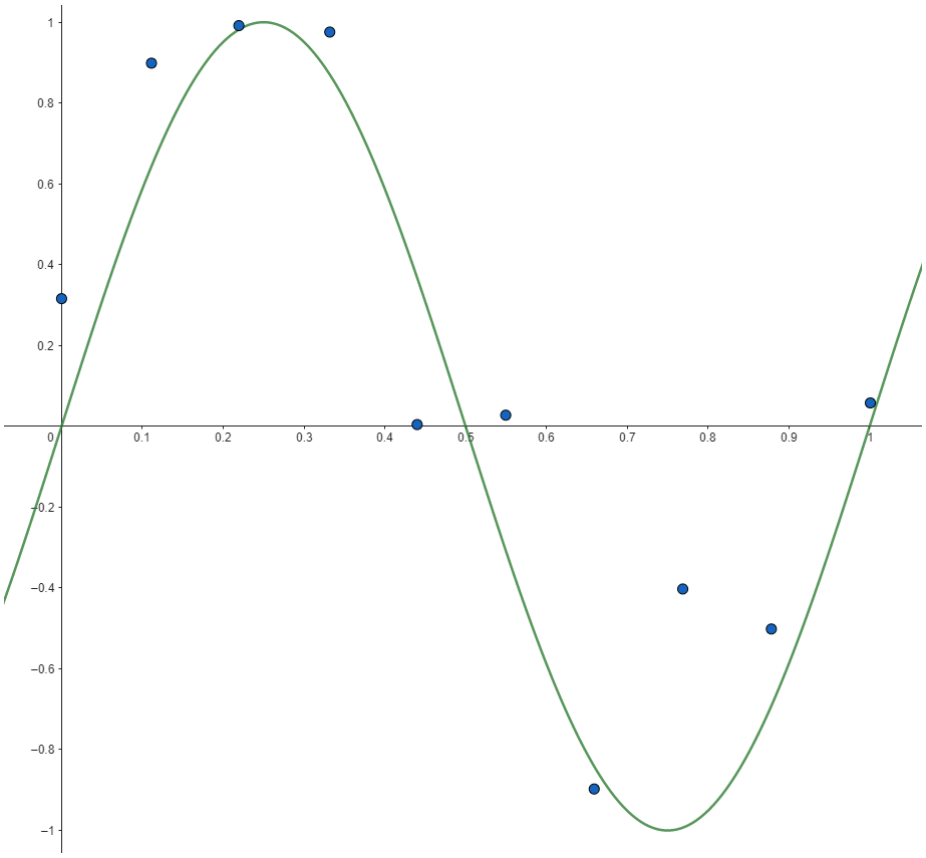

Figure 22: De datapunten met de functie waarop ze gebaseerd zijn

Als we onze lijn niet meer recht hoeven te tekenen, maar een functie met meerdere machten maken, kunnen we wat betere resultaten krijgen. Hieronder zien we de beste lijnen voor de functie $t = a$, $t = ax$, $t = ax^3 + bx^2 + cx + d$ en $t = ax^9 + bx^8 + ... + ix + j$ (figuur **??**)

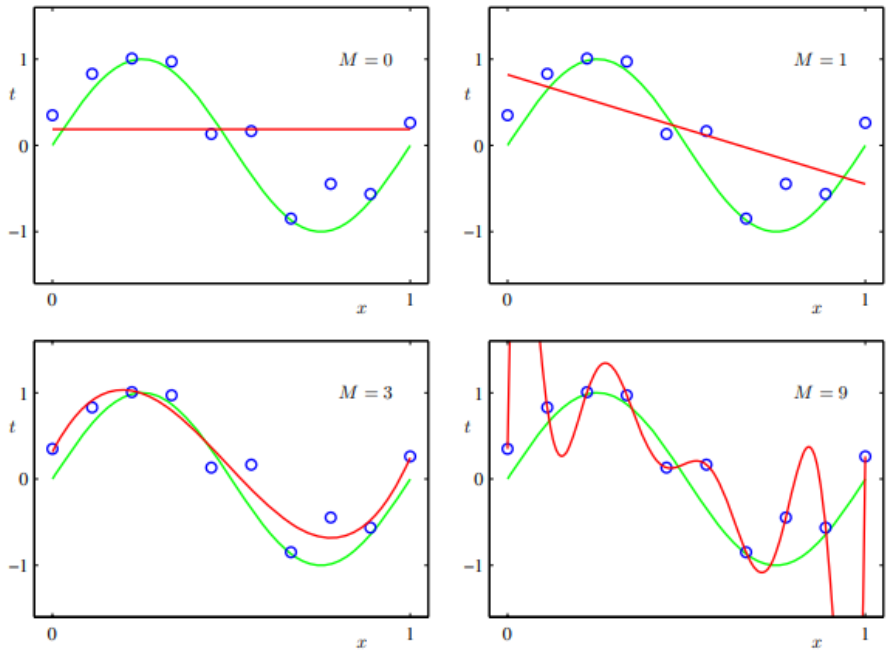

Figure 23: De dataset met verschillende functies erop gepast

Zoals je ziet wordt de error steeds kleiner naarmate je hogere machten aan de functie toevoegt. Tot een bepaald punt, waar de functie zo ingewikkeld wordt dat het precies door elk datapunt kan gaan, maar hierbij ook al het nut als een representatie van de werkelijkheid (hier de groene kromme) verliest. Dit heet overfitten. Dit kan ook gebeuren als een AI te lang op dezelfde dataset zit te trainen.

Nu kan je denken dat je handmatig dit soort problemen kan voorkomen, maar het is niet altijd mogelijk om data zo makkelijk te visualiseren dat er met het blote oog een trend in te vinden is. Bovendien kan een beetje verkeerde lijn op deze data een groot verschil maken als je verder in de toekomst kijkt. Daarom is een goede AI toch de preciezere optie.

Zoals al eerder gemeld is het doel hier om een lijn te trekken die de totale error zo klein mogelijk maakt. Deze error is een functie van de afstand tussen de voorspelling y_i en de werkelijke waarde t_i, maar de precieze functie kan verschillen afhankelijk van wat voor fouten je erger vindt. De error E die wij gebruiken wordt berekend met de volgende functie:

$$E = \frac{1}{2N} \sum_{i=0}^{N} (y_i - t_i)^2 \tag{10}$$

Kijk terug naar de definities (2.4.1) en zie dat y_i een functie is van $\mathbf{x}_i$ en $\mathbf{w}_i$. In dit geval is N elk datapunt, omdat elk punt er wel een beetje naast zit. Tenzij de lijn precies door een punt gaat, maar in dat geval geldt $y_i - t_i = 0$, dus verandert dat punt de weight vector niet in de verkeerde richting.

Nu wordt de weight vector aangepast met de volgende functie:

$$w_{(n)1} = w_{(n)0} - \eta \frac{\delta E}{\delta w_{(n)0}} \tag{11}$$

Merk op dat hier elk coördinaat van de weight vector apart wordt aangepast gebaseerd op de afzonderlijke afgeleiden. Voor je verder gaat, bereken $\frac{\delta E}{\delta w_1}$ om te controleren of die stap je goed afgaat.

B.2.1 Opdracht 2

Om het effect hiervan weer te geven gebruiken wij dezelfde dataset als de vorige les, maar nu is t niet +1 of -1 om voldoende of onvoldoende aan te geven, maar het daadwerkelijk behaalde cijfer. Ongeacht hoe de cijfers liggen kunnen jullie wel zien dat er niet een logische lijn te trekken is in de grafiek van de vorige les die dicht bij alle punten komt om op die manier het cijfer te voorspellen. Daarom breiden wij de dataset uit de vorige les uit met het cijfer dat de leerlingen op de vorige toets van dit vak hebben gehaald als x_3 en hun gemiddelde cijfer van dit vak als x_4. Deze zijn nu alleen weergegeven in tabel vorm, aangezien een grafiek op deze manier 4D zou moeten zijn, wat niet goed weer te geven is.

Gegeven is de weight vector aan het begin $\mathbf{w} = (0, 0.5, -0.5, 0.5, 0.5)$ Bereken nu wat de voorspelde cijfers zijn met behulp van formule **??** en gebruik de uitkomst hiervan met functie **??** om te berekenen hoe de weigth vector wordt aangepast.

	x_1	x_2	x_3	x_4	t
0	0.5	1	7.5	5.7	6.63
1	1	1	8.4	7.5	7.94
2	2	3	3.0	4.1	3.04
3	2	2	5.3	5.9	5.34
4	2.5	4	6.6	5.7	5.69
5	3	5	3.0	2.0	1.90
6	3	3	6.5	5.4	5.76
7	3.5	2	7.0	7.5	7.15
8	3.5	1	7.3	8.0	7.73
9	4	3	5.5	6.0	5.50
10	4	2	6.5	6.9	6.66
11	4	4	4.0	3.5	3.40
12	4.5	3	4.8	5.4	4.89
13	4.5	1	7.2	6.6	7.21
14	4.5	2	6.3	6.5	6.43
15	5	4	4.5	5.2	4.48
16	5	3	6.8	7.2	6.86
17	5	1	8.4	8.0	8.54
18	5.5	3	6.7	7.3	6.89
19	5.5	2	8.3	8.5	8.53
20	5.5	4	5.3	5.5	5.13
21	6	5	4.8	5.3	4.60
22	6	2	7.6	8.0	7.96
23	6.5	4	6.8	6.7	6.61
24	7	3	7.6	7.9	7.82
25	7.5	2	8.5	7.8	8.57
26	8	1	7.5	8.2	8.38
27	9	5	6.5	7.8	6.92
28	9.5	4	8.5	9.0	8.85
29	10	1	9.0	9.1	9.84

Table 4: Data voor het regressie probleem

B.3 Les 3

Deze laatste les wordt er teruggekeken naar het probleem van afgelopen lessen met het voorspellen van het cijfer dat leerlingen gaan halen. In plaats van een half uur met 30 leerlingen om een enkele update uit te voeren op de weight vector, zullen wij nu code gebruiken om tientallen updates per seconde uit te kunnen voeren en op deze manier snel een optimale uitkomst te berekenen.

Jullie hebben een al complete code gekregen en gaan zelf onderzoeken wat het effect is van bepaalde eigenschappen in het programmeren van de vergelijkingen die jullie in de afgelopen lessen geleerd hebben. Dezelfde dataset die jullie gebruikt hebben wordt hier weer gebruikt om de weight vector op te trainen. Een andere dataset met vergelijkbare data wordt gebruikt om de effectiviteit te testen. Dit wordt gedaan om te testen of de AI is geoverfit.

Onderzoek hierbij de volgende eigenschappen:

- Wat voor effect heeft het veranderen van de begin weight vector (W1 voor het classificatie probleem en W2 voor het regressieprobleem in de code)?
- Wat voor effect heeft het veranderen van de leer snelheid η?
- Geeft de kleinste MinError ook de beste uitkomst op de nieuwe dataset in het regressieprobleem?
- Zijn meer iteraties altijd beter?

B.4 Wiskunde cheat sheet

Dit is een cheat sheet voor jullie om te gebruiken in de volgende lessen. Hier staat welke letter waar voor staat, wat het betekent als een letter vet gedrukt is of als er een klein nummertje rechts onder een letter staat, al dan niet met haakjes eromheen. Daarnaast is er ook een korte uitleg hoe vectoren ook al weer met elkaar werden opgeteld, hoe een vector wordt vermenigvuldigd met een constante en wat regels voor afgeleiden.

B.4.1 Definities

In de volgende lessen zullen wij de volgende definities gebruiken:

- Een dik gedrukte letter geeft aan dat deze letter een vector weergeeft.
- $_{(n)}$ geeft een specifiek coördinaat in een vector aan.
- $_i$ geeft een specifiek datapunt in een dataset aan.
- $\mathbf{w}$: Een weight vector. Deze wordt vermenigvuldigd met een input vector om een output te berekenen. Deze vector heeft de vorm $(w_{(0)}, w_{(1)}, w_{(2)}, ..., w_{(n)})$[§§§]. Hier is w_0 de bias. Deze weight vector is ook de lijn die getrokken wordt in de grafiek. Voor een twee dimensionale grafiek heeft dit de vorm $(w_{(0)}, w_{(1)}, w_{(2)})$. Dit kan omgeschreven worden naar de lijnvorm die jullie kennen als $y = -\frac{w_{(1)} * x + w_{(0)}}{w_{(2)}}$. Wat jullie misschien op kan vallen is het feit dat het niet uitmaakt welke specifieke getallen er worden gekozen voor $(w_{(0)}, w_{(1)}, w_{(2)})$, zolang de verhouding tussen die drie maar hetzelfde blijft. Zo geeft bijvoorbeeld $(1, 2, 3)$ precies dezelfde lijn als $(2, 4, 6)$.
- $\mathbf{x}$: Een input vector. Deze wordt vermenigvuldigd met een weight vector om een output te berekenen. Deze vector heeft de vorm $(1, x_{(1)}, x_{(2)}, ..., x_{(n)})$. De 1 hier is toegevoegd om de vermenigvuldiging met de weight vector te laten werken. Deze input zijn de eigenschappen van de data op een rijtje
- y: Een output. Deze wordt gevormd door de volgende berekening:

  $$y = \mathbf{x} \cdot \mathbf{w} = 1 \cdot w_{(0)} + w_{(1)}x_{(1)} + ... + w_{(n)}x_{(n)} \tag{12}$$

  Deze uitkomst die gevormd wordt door het vermenigvuldigen van twee vectoren is dus een getal. Dit getal is in de komende lessen de voorspelling die de AI maakt.
- η: De learning rate. Een waarde tussen 0 en 1 die bepaalt hoe snel de AI de weight vector, en daarmee de voorspellingsmethode, verandert tijdens het leren.
- t_i: De gewenste waarde die uit een voorspelling komt met input $\mathbf{x}_i$. Deze wordt vergeleken met de voorspelde waarde y_i om te kijken hoe goed de code is in voorspellingen maken.

[§§§] Merk op dat de afzonderlijke coördinaten van de vector niet dik gedrukt zijn

- Het optellen van vectoren gaat als volgt: $\mathbf{a}+\mathbf{b} = (a_0+b_0, a_1+b_1, ..., a_(n)+b_(n))$. De uitkomst hiervan is weer een vector.
- Het vermenigvuldigen van een vector met een constante gaat als volgt: $a \cdot \mathbf{b} = (a \cdot b_(0), a \cdot b_{(1)}, ..., a \cdot b_{(n)})$. De uitkomst hiervan is weer een vector.

B.4.2 Afgeleiden

De volgende regels voor afgeleiden zijn handig in de komende lessen:

- $$\begin{aligned} y &= ax_1 + bx_2 \\ \frac{\delta y}{\delta x_1} &= a \end{aligned}$$
- $$\begin{aligned} y &= a(bx_1 + cx_2)^2 \\ &= au^2 \text{ met } u = bx_1 + cx_2 \\ \frac{\delta y}{\delta x_2} &= \frac{\delta y}{\delta u}\frac{\delta u}{\delta x_2} \\ &= 2au \cdot c = 2ac(bx_1 + cx_2) \end{aligned}$$

C Docenten Handleiding

C.1 Algemeen

Deze lessenserie bestaat uit drie lessen, classificatie, regressie en programmeren, met daarvoor eventueel nog wat extra wiskunde om op te halen. De wiskunde die nodig is in de drie lessen is vectoren (het vermenigvuldigen en optellen hiervan) en afgeleiden nemen met behulp van de kettingregel. Hiervoor staat een kleine herinnering in sectie 2.4.1 en **??** voor de leerlingen.

Lessen 1 en 2 introduceren beide een bepaalde vorm van voorspellen. Dit zijn classificatie en regressie respectievelijk. Classificatie is voor het indelen van dingen in groepen, regressie om een specifieke numerieke waarde aan een datapunt te kunnen binden. Hier later meer over bij de uitleg van de specifieke lessen. De leerlingen krijgen die twee lessen eerst een voorbeeld van het soort voorspellen dat er gedaan wordt en spelen zelf de rol van AI door voorspellingslijnen in grafieken te trekken. Hierna worden de leerlingen geïntroduceerd aan de formules die horen bij het leerproces van de AI. Deze formules worden toegepast op een dataset die de leerlingen krijgen. Voor efficiënter werk kan deze dataset verdeeld worden onder de leerlingen, met een datapunt per leerling. Hierna kunnen de leerlingen met hun buren bespreken en elkaars uitkomst controleren. Elke leerling berekend op deze manier de aanpassing die toegepast wordt op de voorspellingslijn. Als alle veranderingen opgeteld worden kan er gevisualiseerd worden wat voor effect deze verandering heeft op de voospellingslijn met het bijgevoegde python bestand.

C.2 Wiskunde

Het kan handig zijn om wat tijd te besteden aan het extra uitleggen van de wiskunde die nodig zal zijn voor de komende lessen. De wiskunde die de leerlingen moeten kunnen is het in-product van vectoren en de kettingregel voor afgeleiden. De doelgroep, leerlingen 5 vwo wiskunde B, zouden al moeten beschikken over deze kennis, maar ik laat het aan de docent over om te bepalen hoeveel tijd er nodig is om deze voorkennis nog extra te activeren. Het is sowieso handig om van tevoren de leerlingen te wijzen op de cheat sheet aan het eind van

Note: In uitgebreidere berekeningen kunnen matrices terugkomen, aan de hand van de voorkennis van 5 vwo leerlingen heb ik besloten om de volgende lessen te beperken tot vectoren. Voor een leerling die met genoeg begrip of een die wiskunde D heeft gevolgd kunnen deze toegevoegd worden. Hierbij zou **w** een matrix worden en **y** een vector zodat er meerdere outputs berekend kunnen worden. Dit zou dan uitgebreid kunnen worden met meerdere stappen inputs en outputs in een neural network. Een voorbeeld hiervan is gegeven in afbeelding **??**. Zo kunnen hier bijvoorbeeld x1 en x2 vectoren van lengte 2 zijn en h3 een weight matrix hebben van 2x3. Door de matrix die bestaat uit x1 en x2 (die is 2x2) te vermenigvuldigen met de weight matrix van h3, komt er een 2x3 matrix uit, wat weer opgesplitst kan worden in twee output vectoren van lengte 3, waarvan de ene naar h1 gaat en de ander naar h2. Op deze manier kunnen hele ingewikkelde modellen gecreëerd worden, zeker als er ook nog non-lineaire functies aan toegevoegd worden (zo kan er bijvoorbeeld een sigmoïde functie toegevoegd worden waarbij de output van h3, in plaats van alleen een lineaire vermenigvuldiging van matrices, ook nog door de functie $y = \frac{1}{1+e^{-x}}$ heen gaat). Dit gaat verder dan de omvang van de lessenserie, maar kan leuk zijn om te vertellen aan een enthousiaste leerling of als leerlingen willen weten hoe de behandelde stof uitgebreid kan worden naar meer ingewikkelde modellen.

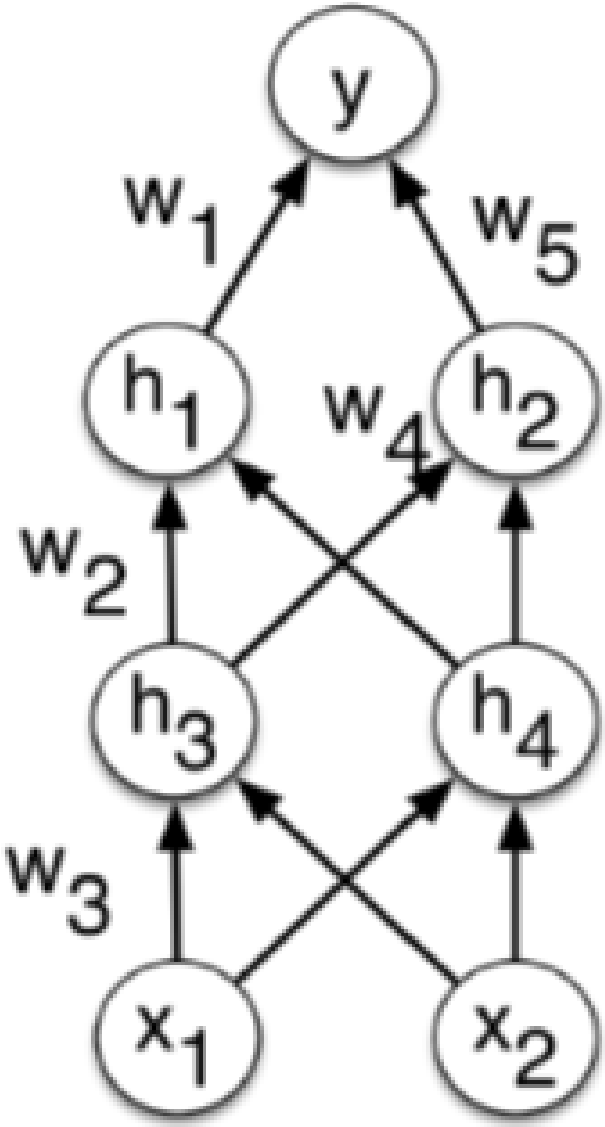

Figure 24: Een simpel Neural Network

C.3 Coderen

C.3.1 Les 1

Het is handig om de eerste les te beginnen met input uit de klas over waar zij aan denken bij AI en Machine Learning. Persoonlijk vind ik een woordwolk met bijvoorbeeld Wooclap een goede manier om dit te doen aangezien dit ook weergeeft als een antwoord vaak gegeven wordt. Hiermee kan de voorkennis van de leerlingen geactiveerd worden en kan een gesprek gestart worden over de invloed van AI op de hedendaagse samenleving in de vorm van bijvoorbeeld ChatGPT en Dall-E. Natuurlijk gaat de les niet zo ver dat ze zelf een nieuwe versie van deze programma's kunnen maken, maar wel de principes die hier ten grondslag aan liggen.

Na deze inleiding begint de uitleg met het verschil tussen classificatie en regressie. Dit kan klassikaal, maar staat ook in het werkblad uitgewerkt. Als je dit klassikaal wil bespreken kan dit ook. Hiervoor de volgende uitleg:

Classificatie wordt gebruikt om AI dingen in groepen te laten verdelen. Een goed voorbeeld hiervan is bijvoorbeeld Dall-E. Dall-E is een code die kunst kan namaken in de stijl van bekende schilders. Hiervoor moet de computer eerst onderscheid leren maken tussen deze verschillende schilders. Dit wordt gedaan door een AI te laten trainen op schilderijen van bijvoorbeeld Rembrandt en van Gogh. Nu verdeelt de AI deze gebaseerd op verschillende eigenschappen en probeert Rembrandt van van Gogh te scheiden gebaseerd op deze eigenschappen. Vervolgens can de code dan omgedraaid worden om deze eigenschappen te gebruiken om kunst te maken. De leerlingen beginnen nu met werken aan het werkblad. Hier wordt uitgelegd wat voor problemen classificatie problemen zijn aan de hand van wat praktische voorbeelden. Hier is het doel dat leerlingen lijnen trekken in grafieken om beide groepen van elkaar te scheiden. Hier is

Zodra de leerlingen een gevoel hebben voor wat het doel is van de classificatie, kunnen ze door naar Opdracht 1. Op het formulier van de leerlingen is een dataset gegeven, zowel visueel als mathematisch. Je kan de leerlingen ieder een punt aangeven, aangezien iedereen 30 punten laten berekenen mij meer tijd lijkt te kosten dan er in de les zit. Hiervan berekenen zij op zichzelf of dit punt goed of fout is geclassificeerd. Hierna kunnen ze in tweetallen hun antwoord bespreken. Tot slot worden alle antwoorden verzameld en kan klassikaal de nieuwe weight vector berekend en bekeken worden. Hieronder is een grafiek toegevoegd waar de nieuwe lijn op aangegeven staat zoals ik die heb berekend met een nieuwe weight vector van [21, 19.05, -31.1] (figuur 14). Als je klassikaal op een andere waarde uitkomt is dit ook te visualiseren met de bijgevoegde code.

Als er nog tijd over is kan je stilstaan bij waarom formule **??** werkt[¶¶¶].

[¶¶¶] Dit kan eventueel gedaan worden met een overdreven voorbeeld waarbij er 100 datapunten die verkeerd ingedeeld worden op 1 plek zitten en het feit dat de lijn door alle plekken gaat met coördinaten met een verhouding $[ax_0, ax_1, ax_c]$. Als die 100 punten verkeerd geclassificeerd zijn, dan, door 100 keer dat ene datapunt bij de originele weight op te tellen, benader je die verhouding $[ax_0, ax_1, ax_c]$ waardoor de lijn verschoven wordt in de juiste richting

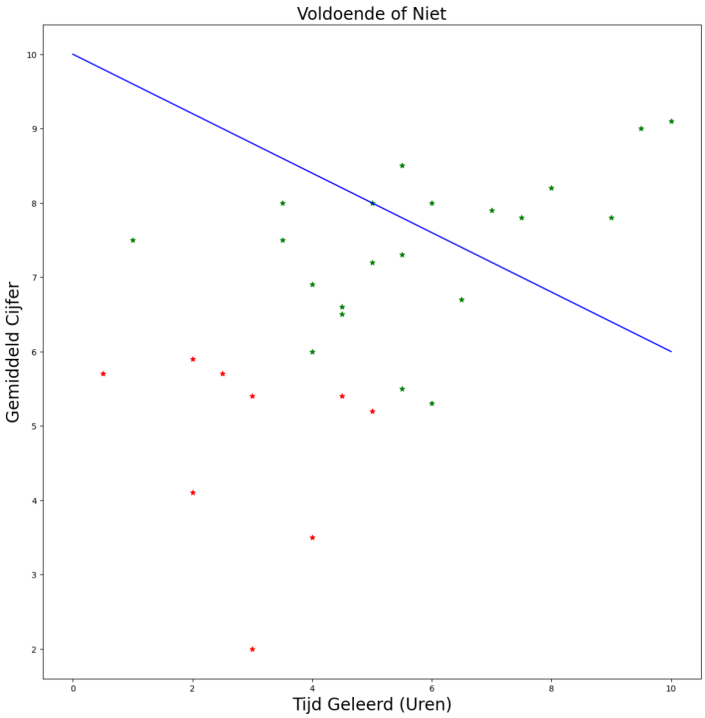

Figure 25: De scheidingslijn aan het begin

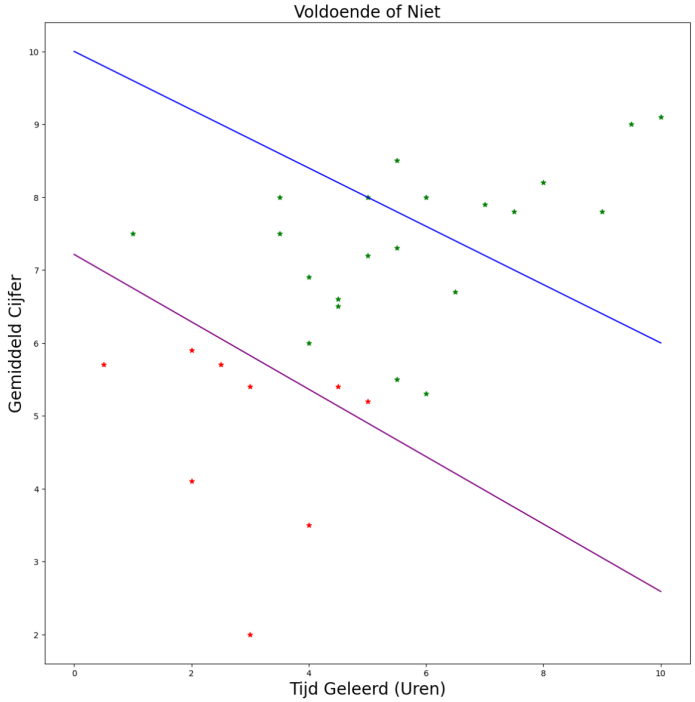

Figure 26: De scheidingslijn na 1 ronde aan veranderingen

Tijdverdeling

- 10 minuten: woordwolk
- 15 minuten: classificatie vs regressie
- 15 minuten: Probleem 1
- 10 minuten: Nabespreken/formule uitleg

Uitleg toevoegen over beweging scheidingslijn door functie Toevoeging meerdere groepen

C.3.2 Les 2

Deze les staat vooral in het teken van een regressie probleem dat voortbouwt op hetzelfde probleem als les 1. Eerst krijgen de leerlingen wederom een paar voorbeelden van regressie problemen. De eerste heeft al een lijn erdoor, de tweede is een voorbeeld waar data gegenereerd is van een sinus en van de leerlingen gevraagd wordt om een lijn hier doorheen te trekken. Dit wordt hierna uitgebreid met een uitleg hoe een vergelijking met een hogere macht een preciezer model kan maken. Echter, als het model te ingewikkeld wordt gemaakt, kan het ook alle representatie van de werkelijkheid (uitgebeeld in figuur **??**). Voordat de leerlingen beginnen aan hun werkblad kan het handig om klassikaal de afgeleide van formule 1 te bespreken. Als je als docent genoeg vertrouwen hebt in het kunnen van je leerlingen kan je deze stap overslaan en de leerlingen ieder voor zich de afgeleide laten berekenen. De afgeleide hiervan naar $w_(n)$ is $\frac{1}{N}\sum_{i=0}^{N}(y_i - t_i)x_{(n)i}$. Op deze manier wordt $\mathbf{w}$ aangepast met $\frac{1}{N}\sum_{i=0}^{N}(y_i - t_i)\mathbf{x}_i$.

Als er tijd is kan er een discussie geopend worden over het belang van de vorm van functie 1 en het gebruik van de afgeleide hiervan. De vorm van de error gaat vooral over het gebruik van het kwadraat van het verschil tussen de voorspelling en de gewenste waarde. Hoe hoger de macht, hoe groter de verandering voor grote fouten. Dit is goed in de zin dat de weight vector sneller aangepast wordt, maar slecht met het risico dat deze aanpassing te snel gebeurt. Ditzelfde geldt overigens voor het kiezen van η. Als de macht van je error of je η te groot is, loop je het risico een grotere error te creëren met het proberen een error weg te werken. Wat de gewenste uitkomst hiervan is, is weergegeven in afbeelding 11. Hier is horizontaal de weight weergegeven, en verticaal de error tussen de voorspelling en de gewenste uitkomst. Idealiter wordt de minimum error met kleine stappen benaderd. De afgeleide geeft aan of de weight groter of kleiner moet zijn, en de error met η geeft aan hoe groot deze verandering wordt. Echter, als deze error te groot wordt berekend, kan het voorkomen dat de weight wel in de juiste richting wordt aangepast, maar zo veel dat de resulterende error groter is dan die waarmee was begonnen. Zie afbeelding 12 voor een visualisatie hiervan. In dezelfde lijn kan een ander gevaar opgebracht worden, namelijk die van overfitten door een te gecompliceerde vergelijking te gebruiken of te lang door te trainen. Zie hiervoor afbeelding **??**. M geeft hier de grootte van de weight matrix en functie aan, in de vorm van $y = \sum_{i=0}^{M}(w_(i)x^i)$ oftewel $M = 2$ geeft de vergelijking $y = w_0 + w_1x + w_2x^2$. De groene lijn geeft de functie aan waar de data uit gegenereerd is (met wat random ruis), de cirkels de datapunten en de rode lijn de uitkomst met de minste error gegeven de functie. Hier is te zien dat naarmate de functie ingewikkelder wordt de error kleiner, tot een bepaald punt waar de error tijdens het trainen 0 wordt (de voorspelling gaat immers precies door alle datapunten), maar er is ook heel goed te zien dat als er nieuwe data voorspeld moet worden deze niet de juiste waardes gaan krijgen.

Tijdverdeling

- 10 minuten: terug kijken naar opdracht 1

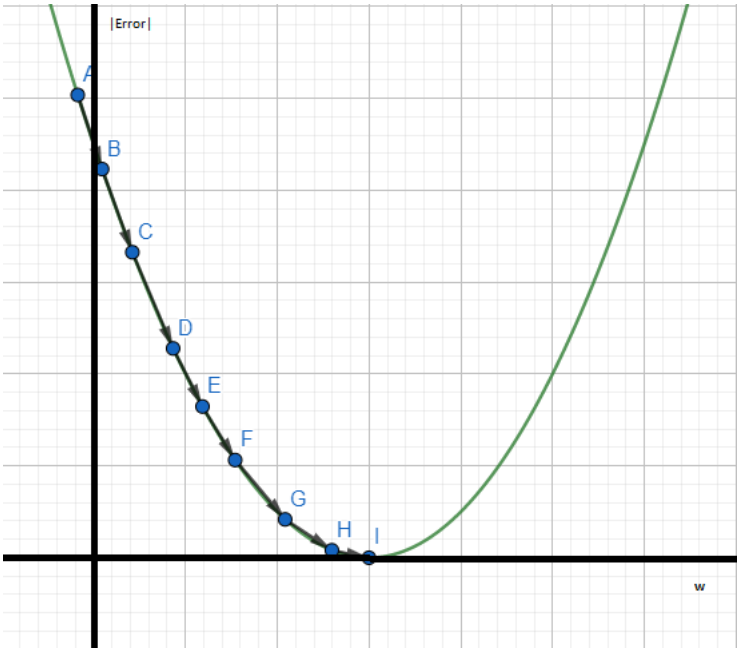

Figure 27: Het gewenste geval

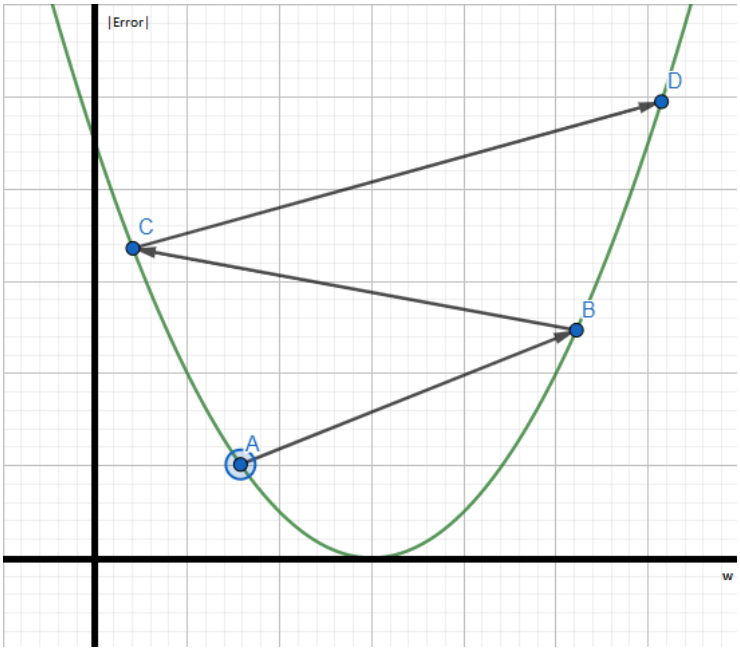

Figure 28: Wat er gebeurt als de weight te snel veranderd

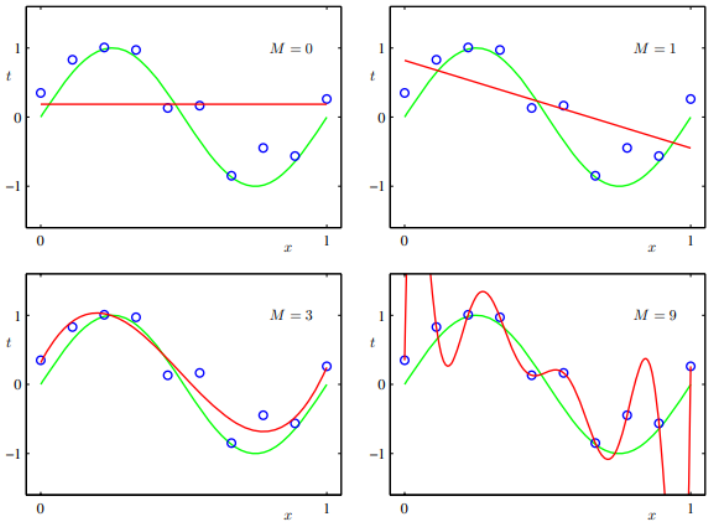

Figure 29: Overfitten

- 10 minuten: Leerlingen zelf update uit laten voeren/bespreken met buren van opdracht 2
- 10 minuten: Klassikaal de update uitvoeren
- 10 minuten: nabespreken
- 10 minuten: Overfitten en de rol van eta bespreken

C.4 Programmeren

De leerlingen krijgen voor deze les de code die automatisch de weight vector traint op dezelfde data die was gebruikt in de afgelopen lessen. Het doel van de les is dat leerlingen het belang van bepaalde stappen gaan onderzoeken. Dit is het aantal updates die gedaan worden en de precisie die hiermee bereikt wordt, het effect van de beginwaarde van de weight vector.

De vindingen die ik verwacht dat de leerlingen zullen vinden is dat met een kleinere waardes in de begin weight vector, er sneller een grote error uit komt en dat eta kleiner moet worden gemaakt. De weight vector heeft kleinere waardes, dus de updates moeten ook kleiner zijn, anders krijg je snel overshoot.

Een andere vinding bij regressie is dat de error bij de test set groter is dan bij de training set. Hoewel dit altijd wel enigszins het geval is kan dit hier een bewijs zijn van overfitten.

Tijdverdeling

- 10 minuten: Ophalen van de twee opdrachten en herhalen van het belang van eta en het aantal genomen stappen
- 30 minuten: Leerlingen gaan zelf aan de slag met code
- 10 minuten: Nabespreken vindingen van leerlingen

D Python Codes

Als deel van de lessenserie zijn ook python bestanden bijgeleverd. Dit zijn drie .ipynb (python notebook) bestanden en een .py (standaard python) bestand. De notebook bestanden zijn gemaakt voor gebruik in de les. De keuze is gemaakt om dit notebook bestanden te maken omdat dit ook naar mijn mening iets makkelijker te gebruiken is. De code voor deze drie zijn hieronder gegeven onder Les 1, 2 en 3. Deze drie notebook bestanden halen de functies uit het standaard python bestand, hieronder gegeven onder Achtergrond bestand. De codes hieronder zijn letterlijk wat er in de python bestanden staat, afgezien van wat extra enters die zijn toegevoegd om alles op de pagina te laten passen. Het resultaat van deze codes zijn vergelijkbaar met figuren 4, 14 en 9, en deze code is ook gebruikt voor het genereren van deze afbeeldingen. Een figuur met alle datapunten, de oude gewichtsvector en de nieuwe gewichtsvector. Soms wordt er ook extra informatie gegeven zoals wat precies de nieuwe vector is, of wat de afwijking is van tevoren en wat het is na het trainen.

Les 1 en 2 zijn alleen voor het gebruik voor de docent. Het doel van deze bestanden zijn puur voor opdracht 1 en 2 uit de eerste twee lessen, voor het geval de leerlingen op een ander getal uitkomen. Hierdoor kan de docent ter plekke redelijk gemakkelijk de nieuwe lijn te laten zien als de berekende vector niet overeenkomt met degene die ik heb voorspeld. Voor gebruiksgemak zijn deze bestanden dan ook zo klein mogelijk gehouden met maar drie regels. Eentje om de functie in te laden, eentje als voorbeeld hoe de berekende vector ingevuld zou moeten worden en eentje om het daadwerkelijk in te vullen.

D.1 Les 1

```
from Achtergrond_Bestanden import *

# RegMan([1,1])
RegMan()
```

D.2 Les 2

```
from Achtergrond_Bestanden import *

#ClassPlot([1,1,1])
ClassPlot()
```

D.3 Les 3

Les 3 is daadwerkelijk voor de leerlingen om te gebruiken. Dit bestaat uit afwisselend stukken code en stukken tekst, hier aangegeven door “BEGIN TEKST” en “EINDE TEKST”, waartussen uitleggende tekst voor de leerlingen staat en

waarbuiten code om te gebruiken voor de leerlingen staat. Wederom begint het met het inladen van de functies, gevolgd door een stuk uitleg over het regressie probleem, de code voor het regressie probleem, uitleg over het classificatie probleem en de code voor het classificatie probleem. De stukken code zijn hier weer zo simpel mogelijk gelaten, met alleen wat regels voor het invullen van een begin vector en alle parameters, gevolgd door een stukje code om daadwerkelijk de functie te laden.

```
#BEGIN TEKST
Dit is het bestand voor de leerlingen voor les 3.
Run eerst het stukje code hieronder om de programma's in te laden
#EINDE TEXT%%
from Achtergrond_Bestanden import *
#BEGIN TEKST%% md
Hieronder kan je nieuwe variabelen invullen voor het
regressieprobleem van Opdracht 1 uit Les 1.
Dit zijn:
-de beginvector bij "W2"
-het maximum aantal updates bij "MaxIter"
-de afwijking waarbij we zeggen dat de lijn goed genoeg is bij "MinAfwijking"
-de afwijking waarbij de code het opgeeft bij "MaxAfwijking"
-de leersnelheid bij "eta".

Voor deze zijn nu allemaal al wat voorbeeld waardes ingevuld.
Deze kan je zelf veranderen.
Let goed op dat decimalen worden aangegeven met een punt,
niet met een komma zoals je gewend bent.
De komma wordt gebruikt om verschillende co rdinaten
        van een vector te scheiden,
net zoals je gewend bent van je GR.
#EINDE TEXT%%
W2 = np.array([1,1], dtype = 'float64')
MaxIter = 10000
MinAfwijking = 0.01
MaxAfwijking = 1000
eta = 0.0001

Regressie(W2, MaxIter, MinAfwijking, MaxAfwijking, eta)
#BEGIN TEKST%% md
Hieronder kan je nieuwe variabelen invullen voor het c
lassificatieprobleem van Opdracht 2 uit Les 2.
Dit zijn:
-de beginvector bij "W1"
-het maximum aantal updates bij "MaxIter"
-de leersnelheid bij "eta".
```

```
Voor deze zijn nu allemaal al wat voorbeeld waardes ingevuld.
Deze kan je zelf veranderen.
Let goed op dat decimalen worden aangegeven met een punt,
niet met een komma zoals je gewend bent.
De komma wordt gebruikt om verschillende co rdinaten
        van een vector te scheiden,
net zoals je gewend bent van je GR.
#EINDE TEXT%%
W1 = [-10,0.5, 1]
MaxIter = 10
eta = 0.01

Classificatie(W1, eta, MaxIter)
```

D.4 Achtergrond bestand

Achtergrond bestanden geeft de functies die de notebook bestanden gebruiken. Dit bestand begint eerst met de dataset die is gebruikt voor de opdrachten. Hieruit worden de behaalde cijfers berekend voor zowel het regressie als het classificatie probleem en aangegeven of het een voldoende is of niet.

```
#%%
import numpy as np
import matplotlib.pyplot as plt
import math

### Dataset Invoeren ###

Labels = ["Tijd Geleerd (Uren)", "Aantal Toetsen",
          "Vorig Cijfer", "Gemiddeld Cijfer"]
Dataset = np.array([
    [0.5, 1, 7.5, 5.7],
    [1, 1, 8.4, 7.5],
    [2, 3, 3.0, 4.1],
    [2, 2, 5.3, 5.9],
    [2.5, 4, 6.6, 5.7],
    [3, 5, 3.0, 2.0],
    [3, 3, 6.5, 5.4],
    [3.5, 2, 7.0, 7.5],
    [3.5, 1, 7.3, 8.0],
    [4, 3, 5.5, 6.0],
    [4, 2, 6.5, 6.9],
    [4, 4, 4.0, 3.5],
    [4.5, 3, 4.8, 5.4],
    [4.5, 1, 7.2, 6.6],
    [4.5, 2, 6.3, 6.5],
```

```
    [5, 4, 4.5, 5.2],
    [5, 3, 6.8, 7.2],
    [5, 1, 8.4, 8.0],
    [5.5, 3, 6.7, 7.3],
    [5.5, 2, 8.3, 8.5],
    [5.5, 4, 5.3, 5.5],
    [6, 5, 4.8, 5.3],
    [6, 2, 7.6, 8.0],
    [6.5, 4, 6.8, 6.7],
    [7, 3, 7.6, 7.9],
    [7.5, 2, 8.5, 7.8],
    [8, 1, 7.5, 8.2],
    [9, 5, 6.5, 7.8],
    [9.5, 4, 8.5, 9.0],
    [10, 1, 9.0, 9.1]])
TestSet = np.array([
    [1, 4, 7.6, 7.9],
    [2, 2, 6.5, 6.3],
    [3, 1, 6.9, 7.2],
    [4, 3, 5.2, 5.0],
    [5, 2, 6.3, 6.7],
    [6, 5, 5.0, 4.8],
    [7, 3, 7.1, 6.9],
    [7, 2, 7.8, 8.2],
    [8.5, 1, 8.4, 8.7],
    [9, 3, 8.3, 8.5]])

### Cijfers berekenen met een formule in een vorm
die overeenkomt met de berekeningen later ###
CijferReg = np.ndarray.round(Dataset[:, 0] * 0.8 + 1.2 + np.random.rand(), 1)
TestCijferReg = np.ndarray.round(TestSet[:, 0] * 0.8 + 1.2 +
        0.8 * np.random.rand(), 1)

CijferClass = np.array([0.2, 0.8]) @
        np.vstack((Dataset[:, 0], Dataset[:, 3]))
TestCijferClass = np.array([0.2, 0.8]) @
        np.vstack((TestSet[:, 0], TestSet[:, 3]))

### Voldoende of onvoldoende target genereren ###

TargetReg = np.zeros(len(CijferReg))
TestTargetReg = np.zeros(len(TestCijferReg))

for i in range(len(CijferReg)):
    if CijferReg[i] < 5.5:
        TargetReg[i] = -1
```

```
    else:
        TargetReg[i] = 1
for i in range(len(TestCijferReg)):
    if TestCijferReg[i] < 5.5:
        TestTargetReg[i] = -1
    else:
        TestTargetReg[i] = 1

TargetClass = np.zeros(len(CijferClass))
TestTargetClass = np.zeros(len(TestCijferClass))

for i in range(len(CijferClass)):
    if CijferClass[i] < 5.5:
        TargetClass[i] = -1
    else:
        TargetClass[i] = 1
for i in range(len(TestCijferClass)):
    if TestCijferClass[i] < 5.5:
        TestTargetClass[i] = -1
    else:
        TestTargetClass[i] = 1
#%%

### Voldoende en onvoldoende opsplitsen in twee aparte datasets
om later in andere kleuren weer te geven###

VoldoendeClass = [[0, 0]]
OnvoldoendeClass = [[0, 0]]
CijferVoldoendeClass = []
CijferOnvoldoendeClass = []
for i in range(len(CijferClass)):
    if CijferClass[i] > 5.4:
        VoldoendeClass = np.append(VoldoendeClass,
                [[Dataset[i, 0], Dataset[i, 3]]], axis=0)
        CijferVoldoendeClass = np.append(CijferVoldoendeClass,
                CijferClass[i])
    else:
        OnvoldoendeClass = np.append(OnvoldoendeClass,
                [[Dataset[i, 0], Dataset[i, 3]]], axis=0)
        CijferOnvoldoendeClass = np.append(CijferOnvoldoendeClass,
                CijferClass[i])
VoldoendeClass = np.delete(VoldoendeClass, 0, 0)
OnvoldoendeClass = np.delete(OnvoldoendeClass, 0, 0)

VoldoendeReg = [[0]]
OnvoldoendeReg = [[0]]
```

```
CijferVoldoendeReg = []
CijferOnvoldoendeReg = []
for i in range(len(CijferReg)):
    if CijferReg[i] > 5.4:
        VoldoendeReg = np.append(VoldoendeReg, [[Dataset[i, 0]]], axis=0)
        CijferVoldoendeReg = np.append(CijferVoldoendeReg, CijferReg[i])
    else:
        OnvoldoendeReg = np.append(OnvoldoendeReg, [[Dataset[i, 0]]], axis=0)
        CijferOnvoldoendeReg = np.append(CijferOnvoldoendeReg, CijferReg[i])
VoldoendeReg = np.delete(VoldoendeReg, 0, 0)
OnvoldoendeReg = np.delete(OnvoldoendeReg, 0, 0)
#%%


### Genereren van beginsituatie classificatie###
def ClassStart():
    W1 = [-100, 4, 10]
    xw = [0, 10]
    yw = [W1[0] / -W1[2], (W1[1] * xw[1] + W1[0]) / -W1[2]]
    plt.figure(figsize=(14, 14))
    plt.scatter(VoldoendeClass[:, 0], VoldoendeClass[:, 3],
            color='green', marker="*")
    plt.scatter(OnvoldoendeClass[:, 0], OnvoldoendeClass[:, 3],
            color='red', marker="*")
    plt.plot(xw, yw, color='blue')
    plt.title("Voldoende of Niet")
    plt.xlabel(Labels[0])
    plt.ylabel(Labels[1])
    plt.show()
    return


### Updaten Classificatie met 1 stap,
        met de optie om handmatig in te voeren ###
### Dit is de code die wordt gebruikt voor Opgave 1 ###
def ClassPlot(manual=[0, 0, 0]):
    W1 = [-100,4,10]
    xw = [0, 10]
    yw = [W1[0] / -W1[2], (W1[1] * xw[1] + W1[0]) / -W1[2]]
    Adjustment = np.array([0, 0, 0], dtype='float64')
    eta = 0.05
    for i in range(len(CijferClass)):
        cords = np.array([1, Dataset[i, 0], Dataset[i, 3]], dtype='float64')
        if W1 @ cords * TargetClass[i] < 0:
            Adjustment += TargetClass[i] * cords
    W1new = W1 + eta * Adjustment
```

```
    if manual != [0, 0, 0]:
        W1new = manual
    ywnew = [W1new[0] / -W1new[2], (W1new[1] * xw[1] + W1new[0]) / -W1new[2]]
    print('De nieuwe gewichts vector hier is', W1new)
    PrintsClass(xw, yw, ywnew)
    return


### Updaten Regressie met 1 stap, met de optie om handmatig in te voeren ###
### Dit is de code die wordt gebruikt voor Opgave 2 ###
def RegMan(manual=[0, 0]):
    W2 = np.array([1, 0.5], dtype='float64')
    eta = 0.001
    dif1 = 0
    deriv = np.array([0, 0], dtype='float64')
    Error = 0
    for i in range(len(CijferReg)):
        cords = np.array([1, Dataset[i, 0]])
        div = 2 * len(CijferReg)
        nom = np.dot(W2, cords) - CijferReg[i]
        Error += np.square(nom) / div
        dif1 += nom
        deriv += np.array([1, Dataset[i, 0]])
    Adjust = (eta * dif1 / len(CijferReg)) * deriv
    W2 -= Adjust
    if manual != [0, 0]:
        W2 = manual
    Errornew = 0
    for i in range(len(CijferReg)):
        cords = np.array([1, Dataset[i, 0]])
        div = 2 * len(CijferReg)
        nom = np.dot(W2, cords) - CijferReg[i]
        Errornew += np.square(nom) / div
        dif1 += nom
        deriv += cords
    ErrorTest = 0
    for i in range(len(TestCijferReg)):
        cords = np.array([1, TestSet[i, 0]])
        div = 2 * len(TestCijferReg)
        nom = np.dot(W2, cords) - TestCijferReg[i]
        ErrorTest += np.square(nom) / div
    print('Oude error was', Error)
    print('Nieuwe error is', Errornew)
    print('Nieuwe gewichts vector is', W2)
    print('Op een nieuwe dataset is de error', ErrorTest)
    PrintsReg(W2)
```

```
    return


### De code om Opgave 1 weer te geven met zowel de oude
        als de nieuwe weight vector ###
def PrintsClass(xw, yw, ywnew):
    plt.figure(figsize=(14, 14))
    plt.scatter(VoldoendeClass[:, 0], VoldoendeClass[:, 1],
            color='green', marker="*")
    plt.scatter(OnvoldoendeClass[:, 0], OnvoldoendeClass[:, 1],
            color='red', marker="*")
    plt.plot(xw, yw, color='blue')
    plt.plot(xw, ywnew, color='purple')
    plt.title("Voldoende of Niet")
    plt.xlabel(Labels[0])
    plt.ylabel(Labels[3])
    plt.show()
    return


def PrintsReg(W2):
    W1 = [1, 0.5]
    xw = [0, 10]
    yw = [W1[0], W1[1] * xw[1] + W1[0]]
    y2 = [W2[0], W2[1] * xw[1] + W2[0]]
    plt.figure(figsize=(14, 14))
    plt.scatter(VoldoendeReg, CijferVoldoendeReg, color='red', marker="*")
    plt.scatter(OnvoldoendeReg, CijferOnvoldoendeReg, color='red', marker="*")
    plt.plot(xw, yw, color='blue')
    plt.plot(xw, y2, color = 'green')
    plt.title("Voldoende of Niet")
    plt.xlabel(Labels[0])
    plt.ylabel("Cijfer")
    plt.show()
    return


### De code om de leerlingen te laten experimenteren
        met het Classificatie probleem ###
def Classificatie(W1, eta, MaxIter):
    xw = [0, 10]
    yw = [W1[0] / -W1[2], (W1[1] * xw[1] + W1[0]) / -W1[2]]
    W1new = W1
    ywnew = [W1new[0] / -W1new[2], (W1new[1] * xw[1] + W1new[0]) / -W1new[2]]
    AantalErrors = 0
    for t in range(MaxIter):
```

```
            W1 = W1new
            Adjustment = np.array([0, 0, 0], dtype='float64')
            AantalErrors = 0
            for i in range(len(CijferClass)):
                cords = np.array([1, Dataset[i, 0], Dataset[i, 3]], dtype='float64')
                if W1 @ cords * TargetClass[i] < 0:
                    Adjustment += TargetClass[i] * cords
                    AantalErrors += 1
            if AantalErrors == 0:
                print('Geen fouten meer gevonden na', t - 1, 'iteraties')
                print('Gevonden gewichtsverctor is', W1new)
                PrintsClass(xw, yw, ywnew)
                break
            else:
                W1new = W1 + eta * Adjustment
                ywnew = [W1new[0] / -W1new[2], (W1new[1] * xw[1] +
                         W1new[0]) / -W1new[2]]
        if t == MaxIter - 1:
            print('Max iteraties bereikt na', t + 1, 'iteraties')
            print('Aantal fouten is', AantalErrors)
            print('Gevonden gewichtsvector zijn', W1new)
            PrintsClass(xw, yw, ywnew)
        return


    ### De code om de leerlingen te laten experimenteren met
            het Regressie probleem ###
    def Regressie(W2, MaxIter, MinError, MaxError, eta):
        Error = 0
        for t in range(MaxIter):
            dif1 = 0
            deriv = np.array([0, 0], dtype='float64')
            if Error < MinError and Error != 0:
                print('Oplossing Gevonden')
                print('Uiteindelijke error is', Error)
                print('Gevonden gewichts vector is', W2)
                print('Aantal iteraties is', t)
                ErrorTest = 0
                for i in range(len(TestCijferReg)):
                    cords = np.array([1, TestSet[i, 0]])
                    div = 2 * len(TestCijferReg)
                    nom = np.dot(W2, cords) - TestCijferReg[i]
                    ErrorTest += np.square(nom) / div
                print('Op een nieuwe dataset is de afwijking', ErrorTest)
                break
            elif Error > MaxError or math.isnan(Error) == True:
```

```
            print('Afwijking te groot na', t + 1, 'iteraties.
                    Hier komt warschijnlijk niets uit.')
            print('Afwijking is', Error)
            break
        else:
            Error = 0
            for i in range(len(CijferReg)):
                cords = np.array([1, Dataset[i, 0]])
                div = 2 * len(CijferReg)
                nom = np.dot(W2, cords) - CijferReg[i]
                Error += np.square(nom) / div
                dif1 += nom
                deriv += cords
            Adjust = (eta * dif1 / len(CijferReg)) * deriv
            W2 -= Adjust
    if t == MaxIter - 1:
        print('Max iteraties bereikt na', t + 1, 'iteraties')
        print('Error op de training is', Error)
        ErrorTest = 0
        for i in range(len(TestCijferReg)):
            cords = np.array([1, TestSet[i, 0]])
            div = 2 * len(TestCijferReg)
            nom = np.dot(W2, cords) - TestCijferReg[i]
            ErrorTest += np.square(nom) / div
        print('Op een nieuwe dataset is de afwijking', ErrorTest)
        print('gewichts vector is', W2)
        PrintsReg(W2)
    return
```