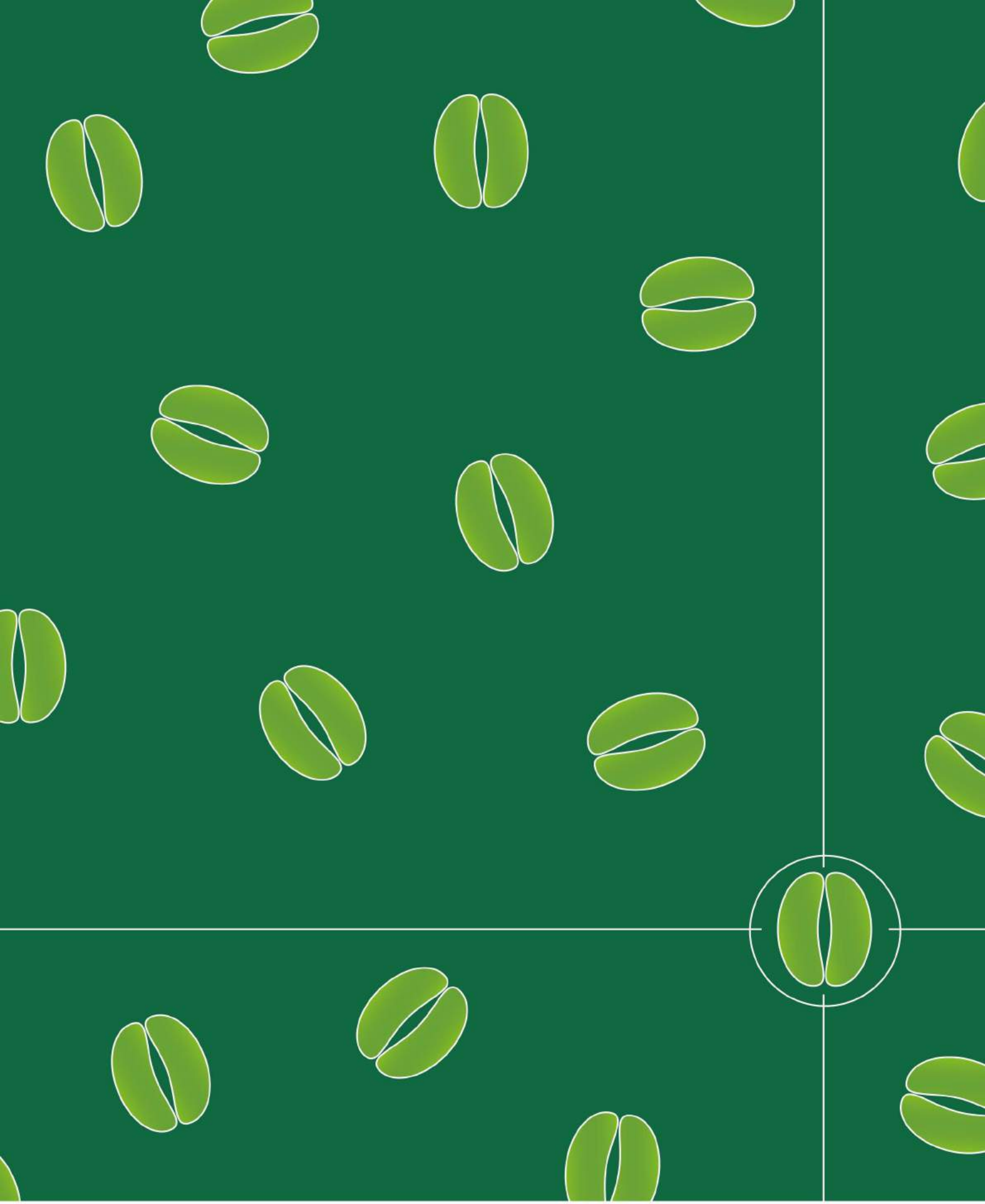


Leerlingencursus

Auteur: Natacha Gesquière
Coauteurs: Sofie Meeus, Jan Van den Bulcke, Francis wyffels

KIKS

Natacha Gesquière

Kunstmatige Intelligentie, Klimaatverandering, Stomata: KIKS

Leerlingencursus

Coauteurs: Sofie Meeus, Jan Van den Bulcke, Francis wyffels

Coauteurs: Sofie Meeus, Jan Van den Bulcke, Francis wyffels

Corrector: Annick Dehennin

Illustrator: Margot De Saegher

KIKS is een Smart Education @ Schools project met steun van de Vlaamse Overheid en imec.

Het KIKS-project is eerste laureaat van de Koningin Paolaprijs voor het Onderwijs 2020.

Deze cursus kwam tot stand met (financiële of logistieke) steun van de Universiteit Gent, Sint-Bavohumaniora, Dwengo vzw, RVO-society, de Provincie Vlaams-Brabant, de Provincie Oost-Vlaanderen, de Vlaamse Overheid en imec.

D/2020/Dwengo/3
ISBN 9789081991797
NUR 950

Eerste druk, oktober 2020

INHOUDSOPGAVE

1 Het KIKS-platform **7**

2 Klimaatverandering **11**

3 Stomata **19**

4 Hoe passen bomen uit het tropisch regenwoud zich aan aan de klimaatverandering? **31**

5 Artificiële intelligentie **37**

6 Digitale beelden **45**

7 Fundamenten van machinaal leren **51**

8 Convoluties **59**

9 ReLU en max pooling **65**

10 Basisconcepten van machinaal leren in de praktijk **69**

11 Implementatie in Keras **99**

HOOFDSTUK
EEN

HET KIKS-PLATFORM

1.1 Interactieve notebooks

Notebooks zijn **digitale documenten** die zowel uitvoerbare code bevatten als tekst, afbeeldingen, video, hyperlinks . . . Via de interactieve notebooks kan je dus nieuwe begrippen aanleren, opdrachten krijgen, tekst typen, foto's bekijken, bepaalde fundamenten van programmeren onder de knie krijgen, aanwezige code uitvoeren en zelf code opstellen.

Jupyter Notebook is een opensourcewebapplicatie waarin de KIKS-notebooks gecreëerd zijn.

De notebooks worden aangeboden via een webserver, toegankelijk via een webbrowser. Je moet op je computer geen extra software installeren om met de notebooks aan de slag te gaan.

De notebooks van KIKS zijn onderverdeeld in drie categorieën:

- basisprincipes van programmeren en beeldverwerking;
- principes van machinaal leren;
- fundamenten van diepe neurale netwerken.

In de notebooks worden **groene, blauwe en gele kaders** gebruikt. De groene vertellen wat de notebook behandelt en leggen linken naar de leerinhouden van het KIKS-project. De blauwe kaders vatten begrippen samen uit de computerwetenschappen. De gele kaders geven tips.

De notebooks bestaan eigenlijk uit een opeenvolging met cellen: **Markdown-cellen** voor tekst en beeld, en **code-cellen** om te programmeren.

Notebook

- Surf naar de notebooks van het KIKS-project via de rechtstreekse link of via de website `https://www.aiopschool.be/kiks`. Klik op `Spawn`.
- In de notebook `WerkenMetNotebooks.ipynb` leer je omgaan met de basisfunctionaliteiten. Doorloop deze notebook.

1.2 Python

Het programmeren gebeurt met Python 3. Python is een zeer toegankelijke programmeertaal, die vaak ook zeer intuïtief is in gebruik.

Python is een **object-georiënteerde taal**. Alles is er een object: elk getal, elk stuk tekst ... Een object neemt een bepaalde plaats in in het geheugen en heeft een bepaalde waarde. Elk object heeft ook een type, bv. integer, string en list. Alle objecten met eenzelfde **type** behoren tot dezelfde klasse. Binnen deze **klasse** zijn er **methodes** voorzien die men specifiek voor objecten van dit type kan oproepen. Naast deze methodes zijn er ook **functies** die men kan gebruiken, zoals `print()`, om iets te laten verschijnen op het scherm, en `input()`, om iets op te vragen aan de gebruiker.

Gaandeweg zal je leren wat de begrippen object, klasse, type, methode en functie inhouden.

Notebook

- Surf naar de notebooks en open de map `IntroductiePython`.
- Zet je eerste stappen in Python met de notebook `AanDeSlag.ipynb`. In deze notebook leer je hoe je tekst laat afdrukken op het scherm, hoe je werkt met variabelen en hoe je input vraagt van een gebruiker. Je werkt met de types string en integer. De concepten sequentie en algoritme worden geduid. Als toemaatje maak je kennis met typecasting.

Bovendien is Python populair omdat er heel wat **modules** voorhanden zijn die je vrij kan gebruiken. Python draagt zo ook bij tot de democratisering van deep learning.

In een module zitten heel wat functies vervat die ervaren informatici reeds hebben geprogrammeerd. Wie dat wilt, kan die modules en de bijbehorende functionaliteiten gebruiken. Daarvoor moet de gewenste module wel geïmporteerd worden in het Python-script.

Een programma in Python noemt men een ***script***.

Notebook

- Leer rekenen in Python met de notebook `Rekenen.ipynb`. In deze notebook werk je met de module `math`. Je maakt kennis met het type floating-point number. Je leert wat een bit en een byte zijn.

1.3 Praktisch

Bij de KIKS-notebooks worden de nodige modules altijd bij het begin van een notebook geïmporteerd.

Tekst wordt ingevoerd met Markdown en wiskundige formules met LaTeX (tussen $$), beide in een Markdown-cel. Code wordt ingevoerd in een code-cel. Cellen kunnen worden bewerkt, verwijderd, toegevoegd, gekopieerd, geknipt en geplakt.

Interessant om te weten is dat in een notebook alle code samenhoort. De notebook onthoudt als het ware welke code reeds werd uitgevoerd, ongeacht in welke volgorde die werd ingetikt in de notebook. **Het is het tijdstip van uitvoeren dat de sequentie van het uiteindelijke script bepaalt.**

Besteed bij het invoeren van code voldoende aandacht aan een leesbare **programmeerstijl** en de nodige verduidelijkende **commentaar**.

1.4 Types

Zoals reeds vermeld is Python een object-georiënteerde taal waarbij elk object een type heeft, bv. integer en string. Maar er zijn ook nog andere types, zoals list en boolean.

Notebook (nu of later)

- Wil je meer weten over de meest courante types? Neem dan de notebook `Datastructuur.ipynb` uit de map `IntroductiePython` door. Je behandelt de types string, integer, boolean, list, tuple, dictionary en NoneType.
- De module `NumPy` wordt veel gebruikt binnen machinaal leren. Deze module laat toe om wetenschappelijke berekeningen uit te voeren en om bv. met matrices te werken. Je leert `NumPy` kennen in de notebook `DatastructuurNumPy.ipynb`. Je leert de ndarray kennen, in het bijzonder de NumPy-lijst.

1.5 Structuren

Misschien ben je erop gebrand om de voornaamste structuren uit programmeertalen te leren kennen. Deze structuren zijn immers dezelfde voor alle programmeertalen, enkel de syntax verschilt.

Deze structuren zijn: de herhalingsstructuur (begrensde herhaling, voorwaardelijke herhaling) en de keuzestructuur.

Notebook - structuren (facultatief)

- De structuren worden uit de doeken gedaan in de notebook `Structuren.ipynb` uit de map `IntroductiePython`.

Samengevat - computerwetenschappen

Notebooks zijn digitale documenten die zowel uitvoerbare code bevatten als tekst, afbeeldingen, hyperlinks . . .

Python is een object-georiënteerde taal. Alles is er een object: elk getal, elk stuk tekst ... Men gebruikt een variabele om naar een object te verwijzen. Een object neemt een bepaalde plaats in in het geheugen en heeft een bepaalde waarde. Elk object heeft ook een type. Alle objecten met eenzelfde type behoren tot dezelfde klasse. Binnen deze klasse zijn er methodes voorzien die men specifiek voor objecten van dit type kan oproepen. Naast deze methodes zijn er ook functies die men kan gebruiken.
Een programma in Python heet een script.
Voor Python zijn er heel wat modules voorhanden die je vrij kan gebruiken.

Na dit hoofdstuk begrijp je waar de volgende begrippen voor staan: variabele, algoritme, sequentie, typecasting, concatenatie, bit, byte, operator.
Je kent de types `string`, `int`, `float`. Je kent de functies `print()` en `input()`.
Facultatief maakte je kennis met andere types zoals `bool`, `list`, `tuple`, `dict` en `NoneType`.
Misschien leerde je ook al dat een matrix in Python wordt voorgesteld door een NumPy ndarray.
Mogelijks leerde je de betekenis kennen van index, herhalingsstructuur (for-lus, while-lus), keuzestructuur (if-elif-else), indentatie en `False` en `True` en gebruikte je methodes zoals `sort()`, `remove()`, `pop()` en `append()` en functies zoals `type()`en `len()`.

HOOFDSTUK
TWEE

KLIMAATVERANDERING

2.1 Paleoklimatologie

Paleoklimatologen bestuderen het klimaat uit (lang) vervlogen tijden. Een schat aan informatie is te vinden in sedimenten op de bodem van de oceanen, is opgeslagen in gletsjers en bewaard in bomen. In die zogenaamde **proxydata** zoals boomringen, ijskernen, en oceaansedimenten maar ook fossiele pollen, rotsformaties in grotten, koralen, vindt men fysieke elementen van de omgeving terug die gecorreleerd zijn met het toenmalige klimaat. Deze natuurlijke proxydata vult men aan met historische data, bijvoorbeeld uit logboeken. Door de gegevens afkomstig van de proxydata te analyseren en te combineren, kan men het klimaat uit een ver verleden reconstrueren (NCEI, 2019).

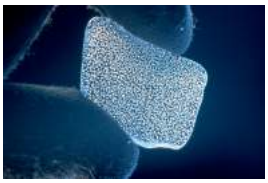
- In het bijzonder geven stomata op soms heel kleine gefossiliseerde bladeren informatie over het CO_2-gehalte in de atmosfeer toen die planten groeiden (zie Figuur 2.1). Zowel het aantal stomata als de grootte van de stomata zijn van belang.
- Bij de polen en hoog in de bergen zijn dikke pakken ijs te vinden. Deze pakken zijn ontstaan door sneeuwval over duizenden jaren. In dat ijs zit nog steeds een deel van de lucht gevangen (Tans, 2018) (zie Figuur 2.2). Men boort diep in het ijs om stof, luchtbellen en zuurstofisotopen, gevangen in het ijs, te onderzoeken en er klimaatomstandigheden van een bepaald tijdperk uit af te leiden. De ijskernen bevatten jaarlijks gevormde lagen, die toelaten de ouderdom van het ijs te bepalen (Boussemaere, 2015). Het ijs verschaft informatie over temperatuur, neerslag, samenstelling van de atmosfeer, vulkanische activiteit en wind.

Met de proxydata heeft men kunnen aantonen dat het klimaat op aarde steeds verandert. Door te begrijpen hoe en waarom het klimaat vroeger geëvolueerd is, ziet men in hoe het huidige klimaat tot stand is gekomen en welke factoren het klimaat kunnen beïnvloeden.

Deze kennis wordt gebruikt om **wiskundige modellen** te bouwen, die tonen hoe veranderende condities op aarde en in de atmosfeer het klimaat de komende decennia kunnen veranderen.

Proxydata zijn gegevens geleverd door een proxy, in dit geval een klimaatproxy. Boussemaere (2015) schrijft: "Een klimaatproxy is een bron aan de hand waarvan men kan inschatten hoe het klimaat er in het verleden uitzag en evolueerde. Een proxy meet de temperatuur of andere gegevens op een indirecte manier".

Figuur 2.1: Gefossiliseerd blad uit de *Fossil Lauraceae leaf database van Nieuw-Zeeland*: goed bewaard blad van de site Grey Lake, behorend tot het geslacht *Cryptocarya* (Steinthorsdottir et al., 2019).

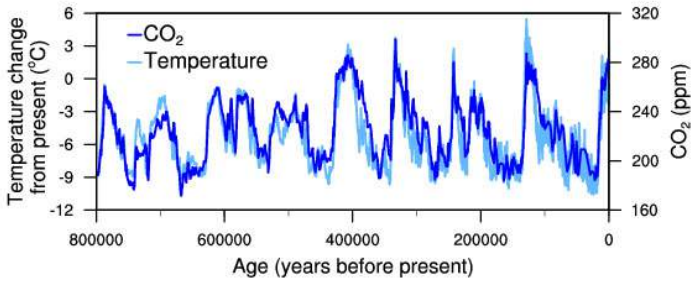


Figuur 2.2: © CSIRO Luchtbellen in ijs (CSIRO. Atmospheric Research, 2019).

Paleoklimatologie kan ook helpen om de invloed van de mens op het klimaat te begrijpen en in te schatten hoe groot die invloed is.

2.2 Atmosferisch CO_2-gehalte en globale temperatuur

Door het klimaat van vroeger te bestuderen, heeft men bv. ontdekt dat het atmosferisch CO_2-gehalte het klimaat beïnvloedt (Maslin, 2014)**.** Men stelde onder andere een opvallend sterke samenhang vast tussen het atmosferisch CO_2-gehalte en de temperatuur van de laatste 800 000 jaar (zie grafiek in Figuur 2.3).



Figuur 2.3: Temperatuursverandering (lichtblauw) en verandering in CO_2 (donkerblauw) gemeten uit de EPICA Dome C ijskern in Antarctica (NOAA, 2008).

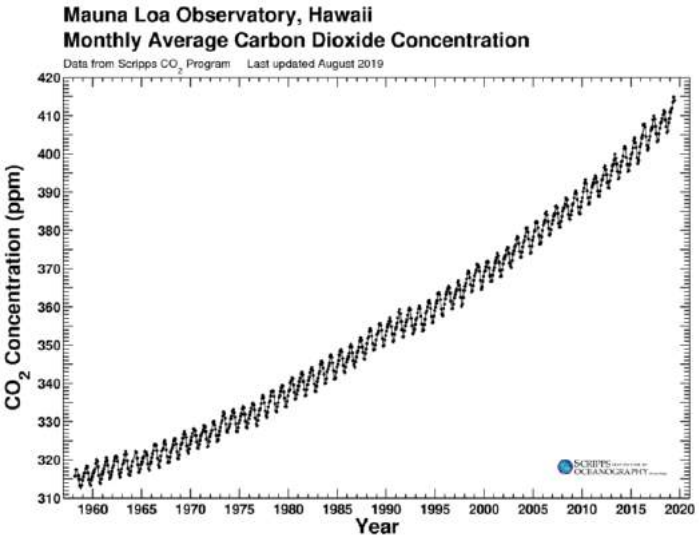
Wetenschapper David Keeling besloot in de jaren 50 om CO_2-metingen te doen, ver van de bewoonde wereld, zodat de meetresultaten niet onbetrouwbaar zouden zijn door CO_2-uitstoot in de buurt (Boussemaere, 2015). In 1958 begon hij met de eerste precisiemetingen van CO_2 op de vulkaan Mauna Loa op Hawaï.

Video

Keeling's Curve. The Story of CO2.
https://youtu.be/0Z8g-smE2sk (American Museum of Natural History, 2014).

De metingen van Keeling vormen samen de **Keelingcurve**. De Keelingcurve is momenteel een van de belangrijkste krommes in het onderzoek naar de klimaatopwarming (Boussemaere, 2015). In Figuur 2.4 valt het schommelende karakter van deze kromme op, naast een onmiskenbaar opwaartse trend. De schommelingen tijdens het jaar zijn er omdat planten tijdens de groei CO_2 opnemen. Die opname is het grootst tijdens de lente.

Momenteel zijn er twee onafhankelijke **meetprogramma's** voor atmosferische CO_2: door het 'Scripps Institute of Oceanography' en door NOAA (Pieter Tans, persoonlijke communicatie). Drie onderzoekscentra, NASA/GISS, HadCRUT en NOAA/NCDC, staan garant voor betrouwbare metingen van de globale temperatuur.

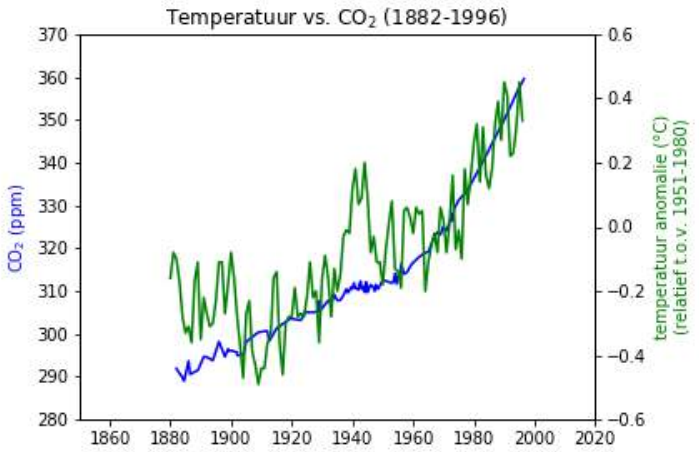


Figuur 2.4: Keelingcurve (Scripps Institution of Oceanography, 2019).

Als de **temperatuurgegevens en de CO_2-data op eenzelfde grafiek** worden uitgezet, bekomt men de grafiek uit Figuur 2.5. De samenhang is onmiskenbaar.

Notebook

- Vooraleer je zelf grafieken kunt maken, bekijk je eerst de notebook `Grafieken.ipynb` in de map `InitiatiePython`.
- Heb je de notebooks `Datastructuur.ipynb` en `DatastructuurNumPy.ipynb` in de map `InitiatiePython` nog niet doorlopen? Doe het dan nu. In de notebook `DatastructuurNumPy.ipynb` leer je bv. werken met een NumPy ndarray, in het bijzonder de NumPy-lijst.



Figuur 2.5: Samenhang verloop temperatuur en CO_2 (1880-1996).

Notebook

- In de notebook `C02.ipynb` zie je hoe de CO_2-concentratie geëvolueerd is over de laatste 2000 jaar en in het bijzonder in de periode vanaf de industriële revolutie tot nu. Je detecteert de kleine ijstijd op deze grafiek.
- In de notebook `C02Temp.ipynb` ontdek je de samenhang tussen het temperatuurverloop en de verandering in CO_2.
- In de notebook `Keeling.ipynb` bekijk je de verandering in CO_2 gedurende een dag en gedurende een jaar. Je begrijpt de link met fotosynthese.
- Vlot het niet zo goed met deze notebooks over CO_2? De notebooks `Datastructuur.ipynb` en `DatastructuurNumPy.ipynb` kunnen helpen.

2.3 Klimaatverandering

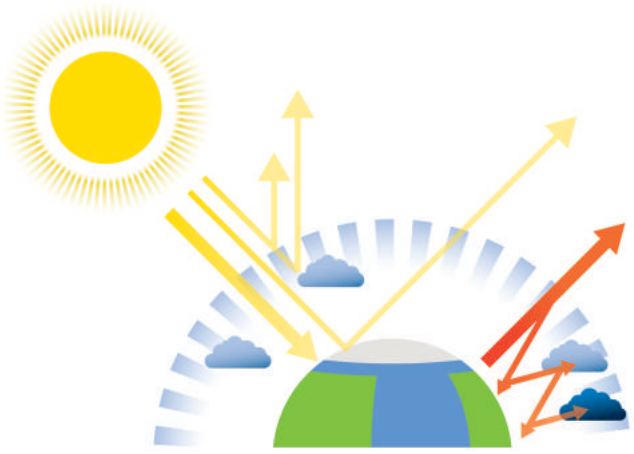
Figuur 2.6: © NASA Atmosfeer (NASA, 2012).

In het milieurapport van de Vlaamse Milieumaatschappij (2019) staat:

"In de atmosfeer zijn gassen aanwezig die de invallende zonnestraling doorlaten, maar de teruggekaatste straling van het opgewarmde aardoppervlak opnemen (zie Figuur 2.7).

Dit fenomeen heet het **broeikaseffect** naar analogie met de werking van glas in een serre. Het leven op aarde dankt zijn bestaan aan dit (natuurlijke) broeikaseffect: de gemiddelde temperatuur op aarde zou anders -18 °C bedragen, in plaats van de huidige +15 °C.

De voornaamste **natuurlijke broeikasgassen** zijn waterdamp (H_2O), koolstofdioxide (CO_2), methaan (CH_4) en lachgas (N_2O). De concentratie van deze gassen in de atmosfeer is het resultaat van talrijke dynamische processen en cycli die op elkaar ingrijpen."



Figuur 2.7: Broeikaseffect.

In dat milieurapport staat ook dat er steeds meer bewijzen zijn dat de temperatuurstijging die we de laatste vijftig jaar waarnemen, grotendeels toe te schrijven is aan menselijke activiteiten: sinds het begin

van het **industriële tijdperk (1750)** is de concentratie van koolstofdioxide, methaan en lachgas in onze atmosfeer sterk toegenomen.

"De verhoogde concentratie van broeikasgassen in de atmosfeer **versterkt** het natuurlijke broeikaseffect en leidt bijgevolg tot een verhoging van de gemiddelde temperatuur en een globale klimaatverandering."

"Men verwacht nog deze eeuw:

- een verhoging van de gemiddelde temperatuur op wereldschaal met 1,1 tot 6,4 °C;
- een toe- of afname van de neerslaghoeveelheden naargelang de regio;
- een stijging van het zeeniveau met 18 tot 59 cm."

Video

Carbon Dioxide Pumphandle 2019.
https://www.esrl.noaa.gov/gmd/ccgg/trends/history.html (CarbonTracker, 2019).

Tijdens de eerste Wereldklimaatconferentie in Genève in 1979 werd de klimaatverandering voor het eerst erkend als een ernstig probleem.

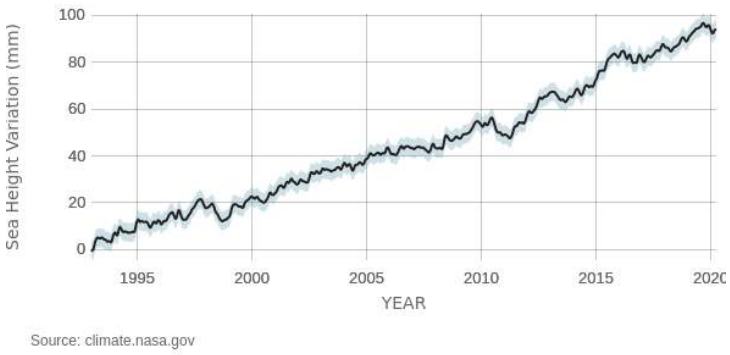
2.4 Afsmeltende gletsjers en stijging van het zeeniveau

Het Intergovernmental Panel on Climate Change (IPCC) heeft getoond dat het globale zeeniveau tussen 1901 en 2010 met 17 tot 21 cm is gestegen. Jaarlijks steeg het globale zeeniveau in die periode met gemiddeld 1,7 mm. De NASA doet de metingen sinds het begin van de jaren 90 met satellieten (zie Figuur 2.8) en vond dat de zeespiegel sinds 1993 met gemiddeld 3,3 mm per jaar stijgt. Dat is meer dan dubbel zo snel als in het begin van de 20ste eeuw (NASA, 2020a). De snelheid waarmee het zeeniveau stijgt, neemt dus toe.

Meerdere factoren spelen hierin een rol. Water dat warmer wordt, zet uit. Hierdoor stijgt de zeespiegel. Ook het afsmelten van ijskappen op Groenland en Antarctica en het terugtrekken van gebergtegletsjers overal ter wereld dragen bij aan de stijging (Maslin, 2014). Het smelten van het zee-ijs, bv. op de Noordpool, draagt niet bij tot de zeespiegelstijging omdat dit ijs al in zee gelegen is.

Notebook (illustratief - facultatief)

- In de notebooks over de gletsjers in de map `IntroductiePython` maak je grafieken die tonen dat de gletsjers veel korter en veel dunner zijn dan pakweg 100 jaar geleden.
  Doorloop eerst de notebook over de Morteratschgletsjer `Morteratschgletsjer.ipynb` en maak erna de opdracht in de notebook van de Silvrettagletsjer `Silvrettagletsjer.ipynb`.



Figuur 2.8: Het zeeniveau geobserveerd door satellieten (NASA, 2020).

Notebook machinaal leren - regressie (facultatief)

- Met machinaal leren kan je bv. classificatieproblemen en regressieproblemen behandelen. Voor het KIKS-netwerk moet je classificatie doorgronden. Maar we willen je de regressie niet onthouden. Zo krijg je een completer beeld van wat machinaal leren inhoudt.
- De notebooks die hieronder worden vermeld, gaan over regressie.
- Het zeeniveau in Oostende wordt sinds 1951 opgemeten. De data zijn terug te vinden op de website van MIRA (Vlaamse Milieumaatschappij, 2019).
  In de notebook `Zeeniveau.ipynb` in de map `IntroductieMachineLearning` gebruik je deze data en ontdek je dat het zeeniveau in Oostende sinds het begin van de metingen al met 13 cm is gestegen. In deze notebook ga je aan de slag met regressie. In deze notebook worden de gegevens gestandaardiseerd. Uitleg over standaardiseren vind je in de notebook `Standaardiseren.ipynb` in dezelfde map.
- Wil je meer notebooks over regressie? Bekijk dan de notebooks over de gletsjers `MorteratschgletsjerRegressie.ipynb` en `SilvrettagletsjerRegressie.ipynb` in de map `IntroductieMachineLearning`, of de notebook `IrisviriginicaRegressie.ipynb` en de notebook `StomataHoogteBomen.ipynb`.

2.5 Consensus

Het IPCC stelt dat de klimaatverandering een feit is en dat er een heel grote consensus is dat de opwarming veroorzaakt is door de uitstoot aan broeikasgassen door de mens (IPCC, 2018)**.**

De klimaatmodellen voorspellen niet 100 % hetzelfde, maar de meeste zijn wel gelijklopend.

Het modelleren van de toekomstige klimaatverandering is niet eenvoudig. Er moeten immers veel factoren in rekening gebracht worden, zoals bevolking, gebruik van energie, uitstoot van broeikasgassen, maar ook gebruik van land en aanwezige **vegetatie**. Vegetatie speelt zelfs een belangrijke rol, aangezien planten de CO_2-cycli van de wereld beïnvloeden (zie hoofdstuk 3).

Samengevat

Dankzij de paleoklimatologie weet men dat de atmosferische CO_2-concentratie het klimaat beïnvloedt. Er is bv. een samenhang tussen het CO_2-gehalte in de atmosfeer en de heersende temperatuur. Tijdens de voorbije 2000 jaar was er eerst een periode met een vrij stabiele CO_2-concentratie met een gemiddelde van 280 ppm, gevolgd door een ongezien snelle stijging sinds de industriële revolutie. Deze stijging kan enkel verklaard worden door de uitstoot van broeikasgassen door de mens. Het heeft bovendien geleid tot een globale klimaatopwarming, iets dat zich de voorbije 2000 jaar nooit eerder heeft voorgedaan.

Planten spelen een grote rol in de CO_2-cycli van de wereld. Om goede klimaatmodellen te kunnen maken, is daarom veel kennis nodig over hoe planten zich aanpassen aan veranderende CO_2-concentraties.

Samengevat - computerwetenschappen

Je leerde wat een puntenwolk is en hoe je een grafiek maakt in Python.

Je werkte vooral met de modules NumPy, pandas en pyplot van Matplotlib.

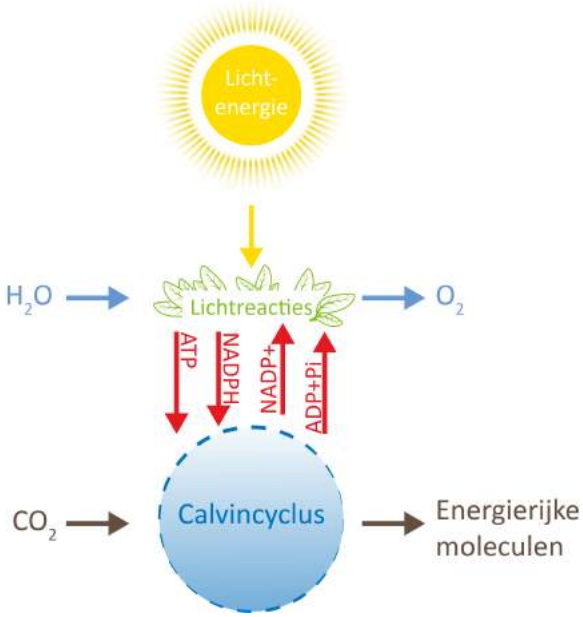
Je weet wat een NumPy ndarray, een index en een csv-bestand zijn. Je kent het verband tussen een ndarray en een matrix.
Je kent de NumPy-functies `where()`, `linspace()`, `arange()`, `append()`, `sum()`, `mean()`, de NumPy-methodes `min()` en `max()`, de pyplot-functies `figure()`, `plot()`, `scatter()`, `show()`, `axis()`, `grid()`, `xlim()`, `ylim()`, `title()`, `xlabel()` en `ylabel()`, de pandas-functie `read_csv()` en de functie `len()`.
In pyplot maakte je gebruik van de parameters `color`, `linewidth`, `linestyle`, `marker`, `xmin`, `xmax`, `ymin` en `ymax`.
Je leerde het type `uint8` kennen.
Je kunt van een matrix de wiskundige dimensie en het aantal elementen, attributen van de ndarray, opvragen met respectievelijk `shape` en `size`.

HOOFDSTUK
DRIE

STOMATA

3.1 Gasuitwisseling tussen een plant en haar omgeving

Zoals alle organismen hebben planten water, mineralen en energierijke koolstofverbindingen nodig om te leven, te groeien en zich te kunnen reproduceren. Planten zijn echter in staat om zelf hun koolstofverbindingen op te bouwen, door met lichtenergie en water koolstofdioxide om te zetten in koolhydraten (glucose). Dit proces heet **fotosynthese** (zie Figuur 3.1).



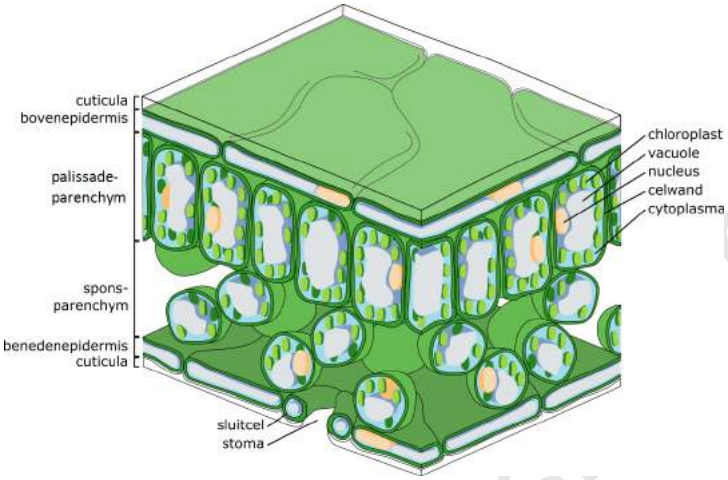
Figuur 3.1: Fotosynthese.

De opname van CO_2 gebeurt vooral langs de bladeren. Om uitdroging tegen te gaan en om de plant te beschermen tegen ziektes, zit er op de buitenkant van het blad een waslaag, de **cuticula**. Die waslaag verhindert echter ook dat er CO_2 in het blad komt. Om de opname van CO_2 toch mogelijk te maken, zijn er microscopisch kleine poortjes in de opperhuid van het blad, de **huidmondjes (stomata)**. Als deze kleine poriën in de opperhuid van het blad open zijn, kan er via die weg CO_2 in het blad opgenomen worden.

Tegelijk kan er ook water uit de plant ontsnappen langs de stomata, men zegt dat de plant transpireert (**transpiratie**). De huidmondjes staan dus in voor afkoeling van de plant en het onderhouden van de sapstroom van wortel tot hoog in de plant. Als het warmer wordt of als de plant genoeg CO_2 heeft opgenomen, zal de plant de huidmondjes sluiten om te voorkomen dat ze te veel water verliest en uitdroogt.

Stomata zijn belangrijk voor de **waterhuishouding** van de plant.

Een plant ademt ook, men spreekt van **respiratie**: 's nachts zet de plant glucose met zuurstofgas om in koolstofdioxide en waterdamp. Hiervoor neemt de plant O_2 op uit de atmosfeer en geeft ze CO_2 vrij. Dit gebeurt ook overdag, maar door de fotosynthese wordt dan meer zuurstof aangemaakt dan ingeademd, waardoor ze bij de fotosynthese O_2 teruggeeft aan de atmosfeer.



Figuur 3.2: Bladstructuur (Zephyris, 2011).

Stomata of huidmondjes zijn dus kleine poriën op bladeren en stengels, die het uitwisselen van gassen (waterdamp, CO_2 en O_2) tussen de plant en haar omgeving mogelijk maken.

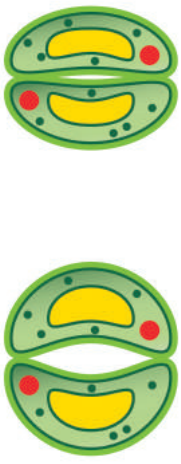
De **stomatale geleidbaarheid** is de mate waarin de gaswisseling tussen een plant en haar omgeving gebeurt.

3.2 Stomata

Een huidmondje ligt tussen twee sluitcellen (zoals te zien op de bladstructuur in Figuur 3.2). Als de sluitcellen in volume toenemen, gaat het huidmondje open. De sluitcellen nemen in volume toe door het opnemen van water. Het gevolg is dat de sluitcellen krommen, waardoor het huidmondje opent (zie Figuur 3.3).

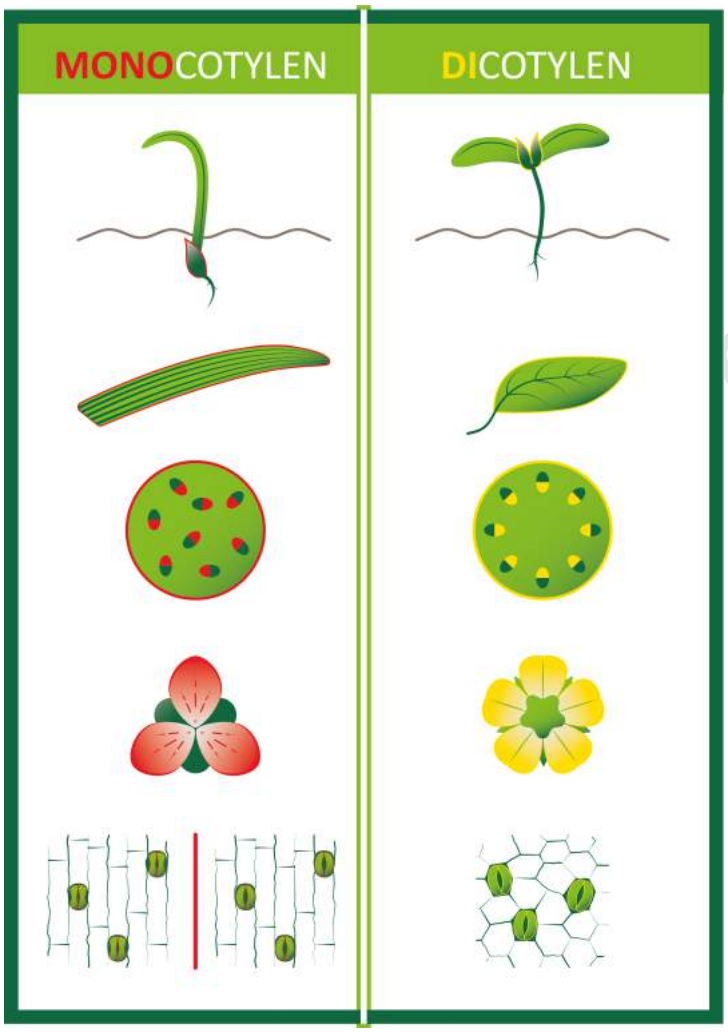
Stomata kunnen op beide zijden van een blad voorkomen (amphistomateus), of op slechts één zijde, meestal de onderkant (hypostomateus), uitzonderlijk de bovenkant (epistomateus), bv. bij een waterlelie.

De stomata en de cuticula worden gezien als **sleutelelementen in de evolutie van planten**. Ze zorgen ervoor dat planten in verschillende en wisselende omstandigheden kunnen leven zonder uit te drogen (Hetherington & Woodward, 2003).



Figuur 3.3: Gesloten en open huidmondje.

Stomata geven informatie over de plaats van de plant in de stamboom van de bloemplanten. Onder de bedektzadigen of bloeiende planten onderscheidt men de **dicotylen** en de **monocotylen** (zie Figuur 3.4). De stomata van dicotylen zijn niervormig, terwijl die van monocotylen niervormig of haltervormig kunnen zijn, afhankelijk van de plant. De nerven van monocotylen liggen evenwijdig en hun huidmondjes ook. De nerven van dicotylen zijn vertakt en hun huidmondjes liggen willekeurig verspreid over het blad.



Figuur 3.4: Monocotylen en dicotylen

Notebook

- In de notebook `Datastructuur.ipynb` in de map `IntroductiePython` komen monocotylen en dicotylen aan bod.

Notebook (nu of later)

- Wil je meer weten over de klassen waartoe objecten behoren? Doorloop dan de notebook `Klassen.ipynb`.

Video

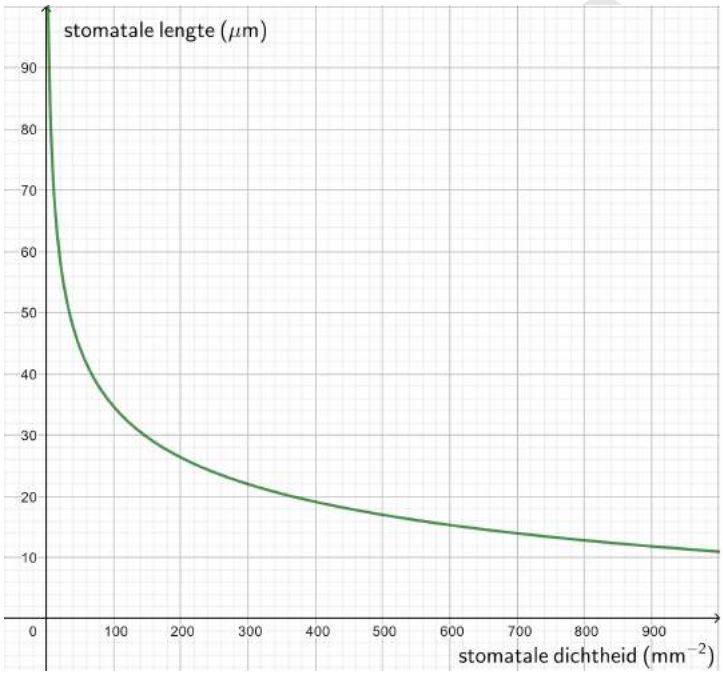
Stomatal Closure in Tradescantia Leaf Cells.
`https://youtu.be/AwyrqfNTuxQ` (davcjal, 2015).

Monocot and Eudicot Germination Time-lapse.
`https://youtu.be/WbG5zu2Vw0I` (Sci- Inspi, 2018).

3.3 Grootte van een stoma en stomatale dichtheid

Afhankelijk van de soort plant en de omgevingsfactoren variëren de grootte van een stoma en het aantal stomata per oppervlakte-eenheid (de **stomatale dichtheid**). Een stoma is tussen de 10 en 100 micrometer lang (vergelijkbaar met de dikte van een mensenhaar) en de dichtheid gaat van 5 tot 1000 stomata per mm² bladoppervlakte.

Een mensenhaar is tussen de 17 en 180 μm dik.



Figuur 3.5: Het verband tussen stomatale lengte (in μm) en stomatale dichtheid (in mm^{-2}) wordt gegeven door: $y = -28{,}75 + 162\,x^{-0{,}2036}$ (Hetherington & Woodward, 2003).

De grootte en de dichtheid van de stomata kunnen verschillende aspecten onthullen van het milieu waarin een plant leefde: temperatuur, atmosfeer, CO_2-concentratie. Dit wordt uitvoerig gebruikt in een paleontologische context om het klimaat waarin de planten groeiden te reconstrueren.

Er is voor meerdere plantenfamilies en fossiele bladeren een duidelijk **verband vastgesteld tussen het aantal stomata en de grootte van de stomata**. Op de grafiek in Figuur 3.5 is te zien dat **hoe meer stomata** er zijn per oppervlakte-eenheid, hoe korter de sluitcellen en dus ook **hoe kleiner de stomata** zijn (Hetherington & Woodward, 2003).

Notebook (nu of later)

- De notebook `Functies.ipynb` in de map `IntroductiePython` bevat een oefening over de grootte van een stoma.

Notebook machinaal leren - regressie (facultatief)

- In de notebook `StomataHoogteBomen.ipynb` in de map `IntroductieMachineLearning` over regressie wordt onderzocht of de hoogte van een boom verband houdt met de grootte van de stomata.

3.4 Openen en sluiten van stomata

Een plant zoekt voortdurend naar een evenwicht tussen vochtverlies en CO_2-opname, afhankelijk van de vochttoestand van de plant, luchtvochtigheid, licht en CO_2-concentratie. Al deze factoren samen met de biologische klok van de plant bepalen of en in welke mate de huidmondjes openstaan.

Stomata zijn onderhevig aan ritmes zoals dag en nacht en de seizoenen.

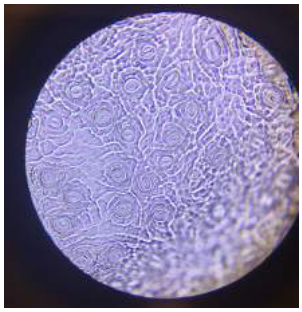
Duidelijk is dat de stomata 'reageren' op **omgevingsfactoren**. Bij eenzelfde plant ziet men soms het verschil tussen bladeren onderaan de plant, die veel beschaduwd zijn, en bladeren in de kruin die veel meer blootgesteld zijn aan het licht. Zelfs op eenzelfde blad kunnen stomata zich anders gedragen omdat dit de plant ten goede kan komen; op eenzelfde blad kunnen sommige stomata open zijn en andere gesloten (zie Figuur 3.6).

Figuur 3.6: Open en gesloten stoma op zelfde blad.

3.5 Microscopie

Niemand heeft reeds huidmondjes gezien met het blote oog. Je kan huidmondjes bewonderen met een **microscoop**. Dat is een toffe ervaring want je kan de huidmondjes ook door de microscoop **fotograferen** met een smartphone. Je kan bv. een monoculaire microscoop gebruiken die 400 keer vergroot.

Als je door de microscoop naar **levend materiaal** kijkt, dan krijg je mooie beelden te zien (zie Figuur 3.7). Daarvoor moet je er wel in



Figuur 3.7: Microfoto van levend materiaal.

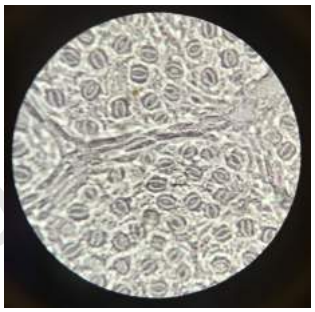
slagen een stuk van de flinterdunne cuticula van het blad te verwijderen. Bij sommige planten lukt dat niet zo goed, bv. door de stugheid van het blad.

Je kan dit opvangen door dezelfde methode te gebruiken als de onderzoekers van de Plantentuin Meise, nl. een **afdruk** nemen van het blad met doorzichtige nagellak (Figuur 3.8).

De foto's zijn dan niet meer zo mooi van kleur, maar grijsachtig (zie Figuur 3.9). Maar al bij al levert het nog steeds mooie plaatjes op; de natuur op zijn best.

Door de planten gevarieerd te kiezen, krijg je ook een **variatie** in de beelden: monocotylen en dicotylen, grote en kleine huidmondjes, stomata van verschillende vormen ...

Figuur 3.8: Bladafdruk nemen met nagellak en plakband.



Figuur 3.9: Microfoto van levend materiaal.

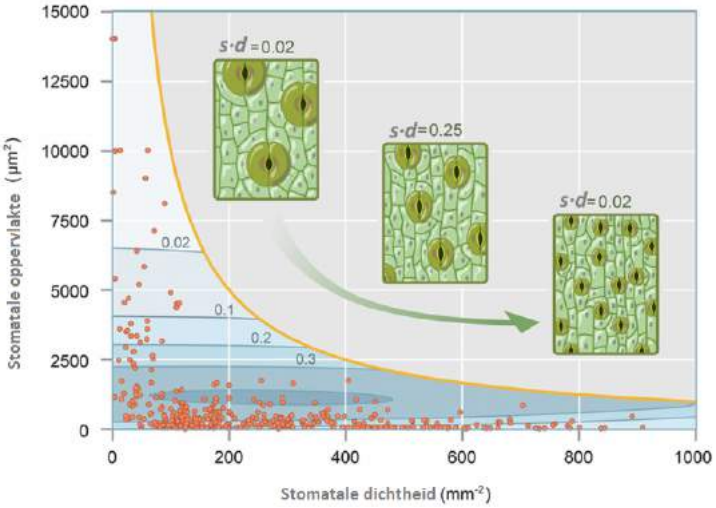
Workshop - microscopie (huidmondjes)

- Verzamel volgroeide bladeren van planten met parallelle nerven en van planten met niet-parallelle nerven.
- Neem eventueel bladeren die veel licht krijgen en bladeren onderaan de plant, bladeren van een struik in de zon en van eenzelfde struik in de schaduw.
- Probeer met een mesje een deel van de flinterdunne cuticula te verwijderen. Breng het aan op een glaasje met een druppel water en dek af met een afdekglaasje.
- Lukt het niet door de textuur van het blad, neem dan een nagellakafdruk van het blad. Breng de afdruk met plakband aan op een glaasje.
- Noteer of je de onderkant of de bovenkant van het blad hebt gebruikt.
- Noteer wanneer en waar je het blad geplukt hebt: seizoen, zon, schaduw ...
- Noteer de naam van de plant.
- Noteer of het een monocotyl of dicotyl is.
- Bekijk het preparaat onder de microscoop.
- Noteer wat je ziet: veel of weinig huidmondjes, grote of kleine huidmondjes, welke vorm ze hebben, hoe ze geordend zijn ...

3.6 Evolutie van planten en hun stomata

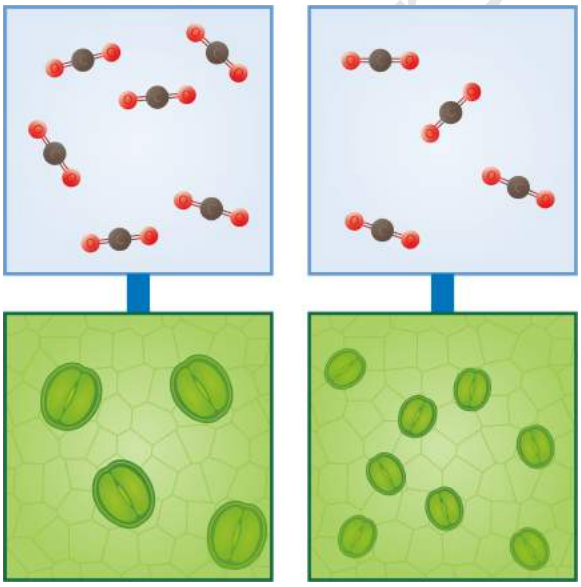
Fossiele bladeren geven veel informatie over de evolutie van planten en hun stomata. Vierhonderd miljoen jaar geleden kwamen stomata bij planten voor het eerst voor. Sindsdien zijn ze geëvolueerd:

er zijn opvallende veranderingen opgetreden waaronder verschillen in de grootte en de dichtheid van de stomata (zie Figuur 3.10) (Hetherington & Woodward, 2003).



Figuur 3.10: Fossil record-based plant leaf stomatal size (*s*) and density (*d*). Hoe meer stomata per oppervlakte-eenheid, hoe kleiner de stomata. De planten zijn zo geevolueerd dat de ingenomen bladoppervlakte (*s.d*) zo klein mogelijk is (Assouline & Or, 2013). Gebaseerd op Franks & Beerling, 2009.

Bij een lage CO_2-concentratie zijn planten met veel stomata in het voordeel en komen ze dan ook veel voor. Wanneer er veel CO_2 in de atmosfeer zit, zijn planten met minder stomata in het voordeel en zullen veel voorkomen (Thanukos, 2018)**.** Dit wordt geïllustreerd in Figuur 3.11.



Figuur 3.11: Verband tussen aantal en grootte van de stomata en de CO_2-concentratie.

Uit fossiel plantenmateriaal blijkt dat planten reageren op langdurige verandering in CO_2 via een aanpassing in de dichtheid van de stomata. Behalve de fotosynthese, zijn ook de transpiratie en de waterhuishouding van de planten factoren die daarin meespelen (Franks & Beerling, 2009; de Boer et al., 2016).

De hoeveelheid CO_2 in de atmosfeer kan daarom afgeleid worden uit de stomatale dichtheid. Aangezien het CO_2-niveau een direct effect heeft op globale temperaturen, geven veranderingen in het CO_2-niveau ook een duidelijk beeld van veranderingen in het klimaat.

Veel planten zijn op zo'n manier geëvolueerd dat de door stomata ingenomen bladoppervlakte (*s.d*) zo klein mogelijk is (Assouline & Or, 2013; Franks & Beerling, 2009)**.**

C_3, C_4 en CAM (facultatief)

De evolutie in het plantenrijk heeft geresulteerd in drie soorten planten: C_3-, C_4- en CAM-planten. Voor elke soort verloopt de fotosynthese anders. In warme gebieden hebben C_3-planten de neiging tot respiratie ten nadele van fotosynthese. C_4- en CAM-planten hebben het fotosyntheseproces aangepast om dit tegen te gaan. Ongeveer 85 % van de plantensoorten zijn C_3-planten, zoals granen en groenten (bv. rijst, graan, spinazie, tomaten) en alle bomen (bv. appelbomen, eik). Ze vormen moleculen met 3 C-atomen tijdens het fotosyntheseproces. Ongeveer 5 % van de plantensoorten zijn C_4-planten, zoals maïs, gierst, suikerriet en vele grassen. Ze maken wel 25 % uit van de planten op aarde en komen vooral voor in tropische, vaak droge gebieden. Ze vormen moleculen met 4 C-atomen tijdens het fotosyntheseproces. De overige 10 % van de plantensoorten zijn CAM-planten, zoals cactus, ananas, Kalanchoë en sedum. Ze komen vooral voor in droge gebieden met hoge dagtemperaturen en lage nachttemperaturen. Ze vormen ook moleculen met 4 C-atomen tijdens het fotosyntheseproces, maar ze verschillen van de C_4-planten omdat ze 's nachts hun stomata openen om CO_2 op te nemen. Zo beschermen ze zichzelf tegen uitdroging. Die CO_2 wordt in de plant opgeslagen en gedurende de dag, als de nodige lichtenergie voorhanden is, terug beschikbaar gemaakt voor de fotosynthese.
(Yamori et al., 2013; Sterrenwacht Armand Pien, 2018).

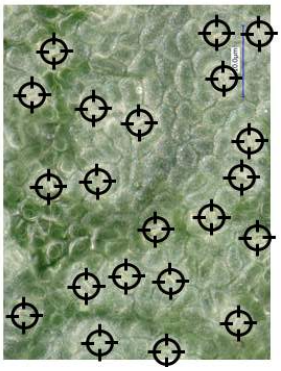
Notebook

- In de notebook `Datastructuur.ipynb` in de map `IntroductiePython` komen C_3-, C_4- en CAM-planten aan bod.
- Voor het KIKS-project is kennis van C_3-, C_4- en CAM-planten facultatief.

3.7 Invloed van de omgeving op de vorming van stomata

Experimenten bevestigen dat dichtheid en grootte kunnen veranderen door genetische aanpassingen, alsook door veranderingen in omgevingsfactoren. Dit betekent dat zowel tussen plantensoorten, tussen individuele exemplaren van eenzelfde soort, als tussen bladeren van eenzelfde plant verschillen kunnen optreden.
Voorbeelden van verandering in omgevingsfactoren:

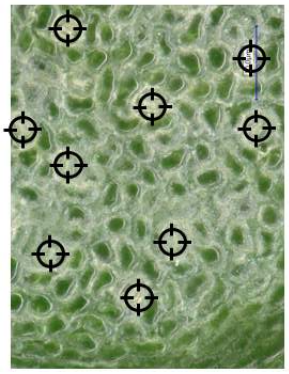
- Bij exemplaren van eenzelfde soort, maar die groeien in verschillende omstandigheden (nl. in de schaduw of in de zon) kunnen verschillen optreden in stomatale dichtheid. Bij een onderzoek op crabwood (*Carapa*) in het Amazonezoud was de stomatale



Figuur 3.12: Bezond blad van *Monodora angolensis*.

dichtheid bij de bezonde planten groter (Camargo & Marenco, 2012).

- Bij eenzelfde plant kan het aantal stomata op beschaduwde bladeren onderaan de plant verschillen van het aantal stomata op bladeren in de kruin die veel meer blootgesteld zijn aan het licht. Sofie Meeus stelde dat vast op de *Monodora angolensis* in de Plantentuin Meise, zoals te zien op Figuren 3.12 en 3.13.



Figuur 3.13: Beschaduwd blad van *Monodora angolensis*.

Notebook - machinaal leren - classificatie (nu of later)

- Doorloop de notebook `StomataZonSchaduw.ipynb` in de map `IntroductieMachineLearning` tot na de animatie. Je ziet dat de planten die in de zon groeiden, gescheiden kunnen worden van de planten die in de schaduw groeiden, en dat louter gebaseerd op de afmetingen en de dichtheid van de huidmondjes.
  Je hoeft nog niet alles te begrijpen in deze notebook.
- Merk je op dat het neurale netwerk eerst volledig willekeurig een rechte kiest en die stap voor stap aanpast tot er een gewenste scheidingsrechte is gevonden?

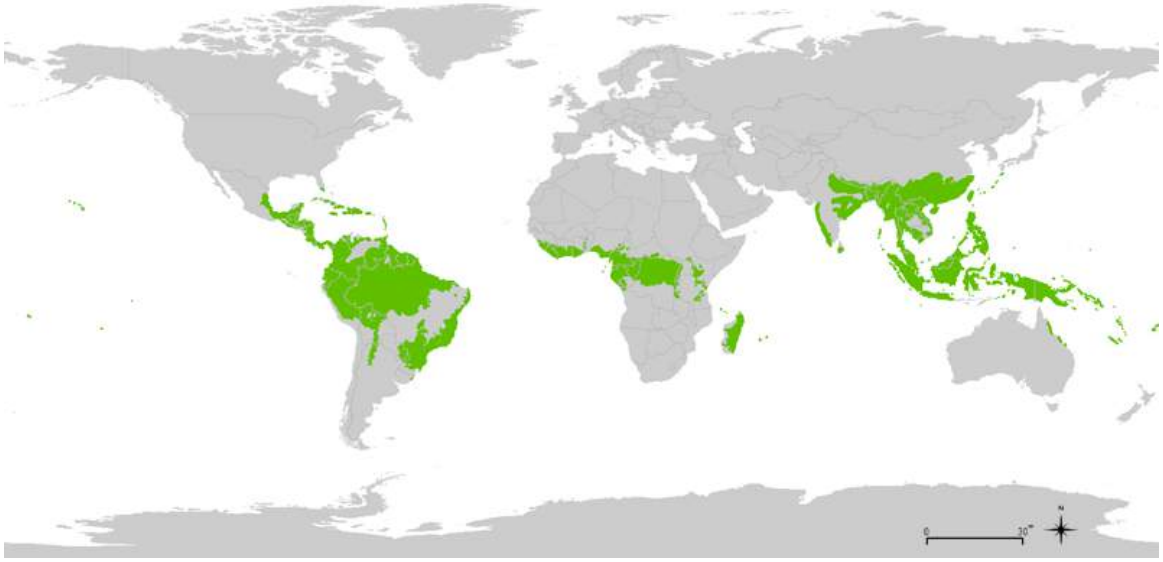
Notebook (facultatief)

- In de notebook `TensorenRgb.ipynb` in de map `IntroductiePython` worden deze foto's van Sofie Meeus en microfoto's van de Plantentuin Meise gebruikt.

3.8 Stomata en klimaatverandering

De water- en CO_2-cycli van de wereld worden beïnvloed door de transpiratie van planten op het land. Men kan dus zeggen dat stomata een impact hebben die globaal is, een impact op de hele wereld. Er wordt het meest getranspireerd door de vegetatie in de warme beboste gebieden van de tropen (zie Figuur 3.14) (Hetherington & Woodward, 2003). De tropische regenwouden situeren zich in gebieden in de buurt van de evenaar. Ze bevinden zich in Centraal- en Zuid-Amerika, in Afrika, in Azië en op de eilanden rond Australië. **Het is belangrijk om te begrijpen hoe planten deze transpiratie zullen aanpassen aan de huidige klimaatverandering.** Wetenschappers onderzoeken ook hoe de huidige klimaatverandering de vorming van stomata beïnvloedt.

Een veranderend klimaat heeft een impact op de **biodiversiteit** en omgekeerd kan de biodiversiteit ook de klimaatverandering beïnvloeden. Bv. transpiratie verkoelt niet enkel de plant, maar ook haar



Figuur 3.14: De tropische regenwouden zijn groen gekleurd op de kaart.

omgeving. Er zijn grote verschillen in hoe planten reageren op de CO_2-toename en in de mate waarin planten CO_2 opnemen.

Het is bv. cruciaal dat men inzicht krijgt in de aanpassingen van gewassen om de **voedselvoorziening** te kunnen bewaken.

Video

Stomata and global climate cycles.
`https://youtu.be/eD2J3PBoERI` (Bergmann, 2015).

Samengevat

Planten passen zich aan aan veranderende omstandigheden, zoals licht, seizoen, temperatuur, watervoorraad en de atmosferische CO_2-concentratie. Via hun huidmondjes regelen ze hun waterhuishouding en zorgen ze ervoor dat ze CO_2 opnemen om te groeien.
Doorheen de evolutie hebben planten hierin een evenwicht gevonden. Hoe hoger de CO_2-concentratie, hoe minder huidmondjes. Hoe groter de stomatale dichtheid, hoe kleiner de huidmondjes. Daarom kan men de dichtheid en de afmetingen van huidmondjes gebruiken als tool om het heersende klimaat waarin de planten groeiden, te reconstrueren.

Het is belangrijk om te weten hoe snel planten zich aanpassen, om bv. te kunnen inschatten hoe de biodiversiteit en de voedselvoorziening beïnvloed zullen worden door een toenemend CO_2-gehalte. Daarom is het interessant om te onderzoeken of de verandering van CO_2-concentratie tijdens de voorbije 100 jaar reeds een meetbaar effect teweegbracht op de stomata van planten.

Na dit hoofdstuk weet je wat het verband is tussen fotosynthese en stomata.
Je weet wat monocotylen en dicotylen zijn.
Je weet dat de CO_2-concentratie en andere omgevingsfactoren een invloed hebben op de vorming van huidmondjes.

Samengevat - computerwetenschappen

Misschien hebben functies en tensoren in Python nu geen geheimen meer voor je.

HOOFDSTUK
VIER

HOE PASSEN BOMEN UIT HET TROPISCH REGENWOUD ZICH AAN AAN DE KLIMAATVERANDERING?

4.1 Het onderzoek

Wetenschappers van de Plantentuin Meise en de UGent onderzoeken hoe bomen uit het tropisch regenwoud zich aanpassen aan de klimaatverandering. De vorming van stomata op hun bladeren is immers gevoelig voor de CO_2-concentratie in de atmosfeer. De onderzoekers tellen het aantal stomata op de bladeren en ze meten hun grootte op. Erna vergelijken ze de resultaten van recent materiaal met die van materiaal van honderd jaar terug.

Het tellen en meten van stomata is echter een heel tijdrovende bezigheid. **Artificiële intelligentie** biedt hierop een antwoord: een **convolutioneel neuraal netwerk** is uitermate geschikt om beelden, en dus ook huidmondjes op bladeren, te herkennen.

Het team van de UGent en de Plantentuin Meise bouwt zulke neurale netwerken.

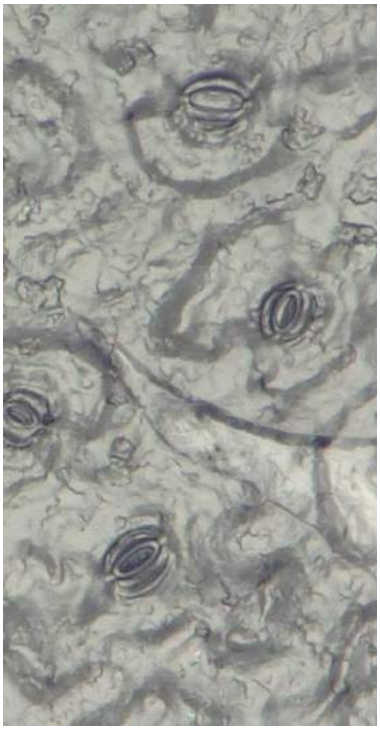
KIKS-animatiefilm

- Op de website `https://www.aiopschool.be/kiks` kun je het verhaal van KIKS bekijken in een animatiefilmpje.

4.2 Verzamelen van de dataset

Stomata zijn microscopisch kleine onderdelen van een plant, ze zijn tussen 10 en 100 micron lang. De dataset bestaat uit foto's van bladeren genomen met een microscoop. In het vervolg spreken we van **microfoto's** (zie Figuur 4.1).

Om de trainingset te genereren gebruikten de onderzoekers **19 gewone tropische boomsoorten** die behoren tot 12 families van bloeiende planten. Voor elk van de 19 soorten namen ze enkele stalen voor elke tijdsperiode.



Figuur 4.1: Foto uit de KIKS-dataset: *Coffea boiviniana Capuron*.

De kwaliteit van gedroogd materiaal is minder goed dan dat van

vers materiaal. Zelfs met een goede microscoop zijn de stomata op gedroogd materiaal niet altijd goed te zien, zeker ook omdat het materiaal vaak oud en broos is. Daarom werd van elk blad een **afdruk** genomen met **transparante nagellak**, wat een betere kwaliteit van foto's gaf (zie Figuur 4.2).

Eens de nagellak opgedroogd was, werden de afdrukken met dubbelzijdig plakband op een objectglaasje overgebracht met de afdruk naar boven. Met behulp van een digitale microscoop werd van drie zichtvelden per blad een foto genomen, maar dan duizendmaal vergroot. **Zo'n foto heeft een formaat van 1600 op 1200 pixels.**

Figuur 4.2: Bladafdruk nemen met nagellak en plakband.

Voor het onderzoek was het nodig dat het netwerk de (volledige) huidmondjes per zichtveld kon tellen. Deze gegevens werden omgezet naar het aantal huidmondjes per vierkante millimeter (een zichtveld heeft een oppervlakte van 0,09 mm^2), zodat men dus de stomatale dichtheid bekwam.

Een zichtveld is 344 μm op 258 μm.

4.3 Voorbereiden van de dataset

Voordat het netwerk de huidmondjes kan tellen of meten, moet hij ze eerst **herkennen**.

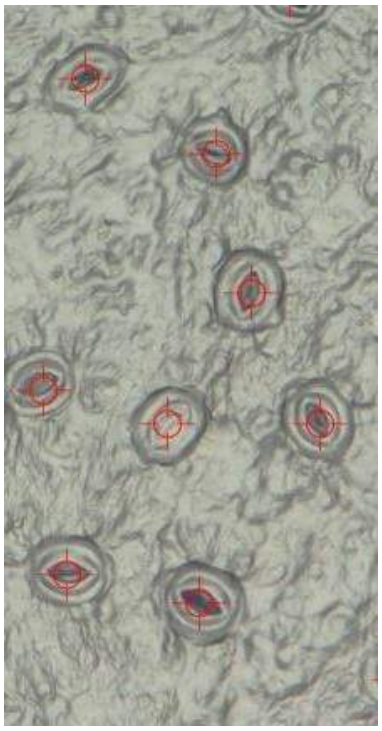
Om zo'n netwerk te trainen in het detecteren van huidmondjes, toont men heel veel voorbeelden aan het netwerk: foto's van stomata en foto's van delen van bladeren waarop zich geen stoma bevindt.

Om een netwerk te trainen in het meten van de stomata, toont men voorbeelden van bladeren waarop de afmetingen van de aanwezige stomata zijn aangeduid.

Op voldoende microfoto's werd het midden van elke stoma manueel aangeduid door een mens. Men noemt dit **annoteren**. Het resultaat is een geannoteerde foto zoals in Figuur 4.3.

Workshop - zelf annoteren

- Je kan zelf huidmondjes aanduiden op foto's, online via `https://aiopschool.be/kiks/annoteren`.



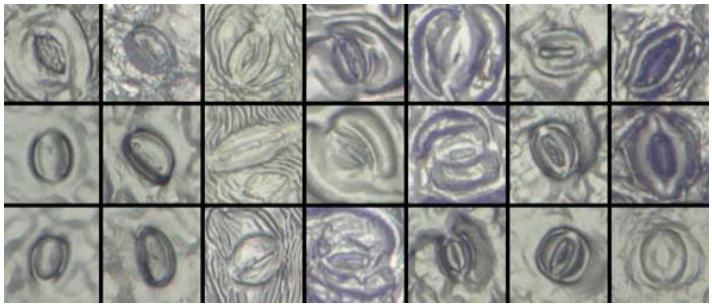
Figuur 4.3: Geannoteerde foto van *Entandrophragma utile*.

Uit deze geannoteerde foto's werden voorbeelden voor het netwerk gegenereerd: er werden stukken van 120 op 120 pixels uit de foto's geknipt; sommige met een stoma in het midden, sommige met een stoma niet in het midden, andere zonder stoma. Zo bekwam men de voorbeelden van stomata en de voorbeelden van delen van bladeren zonder stomata (zie Figuren 4.4 en 4.5).

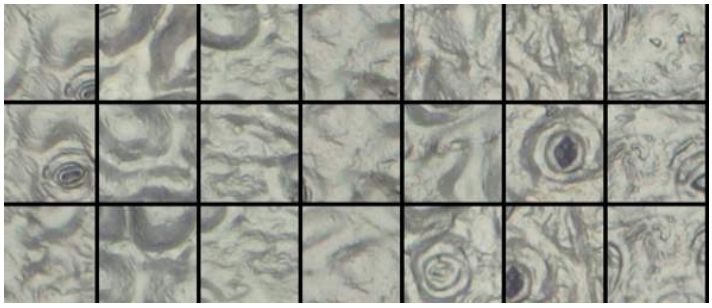
Al deze afbeeldingen van 120 op 120 pixels werden dan **gelabeld** aan de dataset toegevoegd, 12 000 voorbeelden van stomata en 72 000 voorbeelden van delen van blad zonder stoma.

Vervolgens gebruikte men die gelabelde voorbeelden om een diep neuraal netwerk voor het detecteren van stomata te trainen (zie

hoofdstuk 7).



Figuur 4.4: Voorbeelden van stomata.



Figuur 4.5: Voorbeelden zonder stomata.

Voor het trainen van het neurale netwerk beschikte men over een **trainingset** die bestond uit foto's met enkel niervormige stomata. Daarnaast was er ook nog een **testset**, met behalve niervormige ook haltervormige huidmondjes. De testset werd opgebouwd uit 16 andere soorten tropische bomen van 7 families.

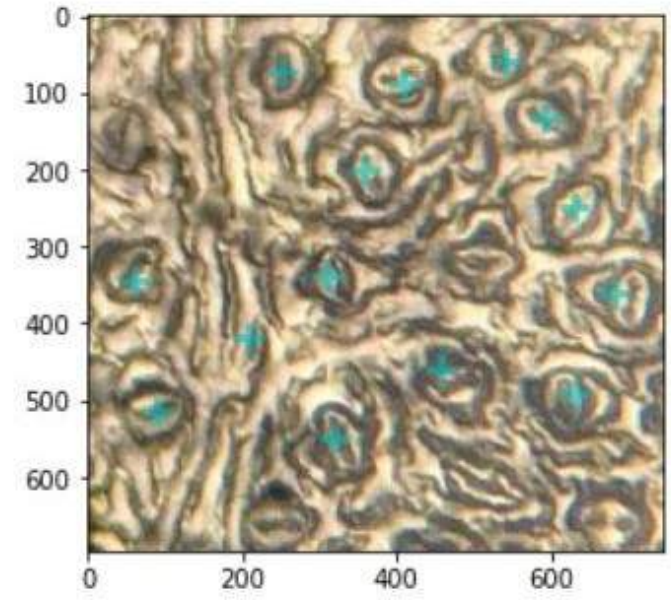
Notebook

- In puntje 2 van de notebook `StomataDetectie.ipynb` in de map `IntroductieDeepLearning` zie je hoe het netwerk te werk gaat om de huidmondjes te tellen.

4.4 Huidmondjes tellen op eigen microfoto's

Workshop - tellen van huidmondjes

- KIKS beschikt over een notebook om huidmondjes op foto's te laten tellen: `EigenStomataDetectie.ipynb` in de map `IntroductieDeepLearning`.
- Voer je foto's in in die notebook.
- De code in de notebook geeft een uitvoer zoals in Figuur 4.6.
- Merk op dat bij sommige foto's het tellen goed lukt, maar bij andere foto's minder.
- Kan je dit verklaren? Denk hiervoor aan: onscherpe foto's, te kleine huidmondjes, moeilijk te herkennen huidmondjes, te veel kleuren in de foto's ...



Figuur 4.6: Getelde stomata op bladafdruk van klimop.

De redenen waarom het tellen soms niet goed lukt, staan in rechtstreeks verband met de ***bias*** waarmee men in deep learning-systemen te kampen heeft.

Over bias lees je in paragraaf 5.5.

Deze problemen ontstaan omdat het netwerk enkel geschikt is voor foto's die gelijkaardig zijn aan de foto's waarmee het netwerk getraind werd.

Samengevat

Wetenschappers van de Plantentuin Meise en de UGent onderzoeken hoe bomen uit het tropisch regenwoud zich aanpassen aan de klimaatverandering, in het bijzonder aan de toename van de CO_2-concentratie in de atmosfeer. De onderzoekers tellen het aantal stomata op de bladeren en ze meten hun grootte op. Erna vergelijken ze de resultaten van recent materiaal met die van materiaal van honderd jaar terug.
De stomata worden geteld op microfoto's van nagellakafdrukken van de bladeren. Het tellen is geautomatiseerd met een diep neuraal netwerk, dus gebruikmakend van artificiële intelligentie.
Zo'n neuraal netwerk wordt getraind met gelabelde voorbeelden.

Samengevat - computerwetenschappen

Je weet wat de volgende begrippen betekenen: dataset, annoteren, trainingset, gelabelde voorbeelden.

Stand van zaken van het wetenschappelijk onderzoek (2020)

Uit het onderzoek blijkt dat de bomen uit het tropisch regenwoud in Afrika zich inderdaad hebben aangepast aan de toename van atmosferische CO_2, maar toch niet helemaal zoals op voorhand ingeschat.
De stomatale dichtheid is afgenomen, zoals verwacht, maar op deze relatief korte termijn van 100 jaar zijn de huidmondjes (nog) niet groter geworden. Het ziet ernaar uit dat de toename van atmosferische CO_2 toch niet geleid heeft tot meer fotosynthese. De bomen zijn er ook niet in geslaagd om efficiënter met hun watervoorraad om te springen (Bauters et al., 2020).

HOOFDSTUK
VIJF

ARTIFICIËLE INTELLIGENTIE

5.1 Wat is AI?

Kunstmatige (KI) of artificiële intelligentie (AI) komt volop aan bod in de media. Populaire thema's zijn: chatbots, zelfrijdende auto's, virtuele assistenten en het verlies aan jobs.

AI

De Europese Commissie hanteerde in 2018 de volgende definitie voor AI (PwC, 2018): "Kunstmatige intelligentie verwijst naar systemen die intelligent gedrag vertonen door hun omgeving te analyseren en – in zekere mate zelfstandig – actie te ondernemen om specifieke doelstellingen te verwezenlijken. Op KI gebaseerde systemen kunnen louter softwarematig zijn en actief zijn in de virtuele wereld (bijvoorbeeld stemgestuurde assistenten, software voor beeldanalyse, zoekmachines en systemen voor spraak en gezichtsherkenning), maar KI kan ook in hardware-apparaten worden geïntegreerd (bijvoorbeeld geavanceerde robots, zelfrijdende auto's, drones of toepassingen van het internet der dingen)".
Naast deze definitie worden nog andere definities geponeerd. Tegmark (2017) zegt eenvoudigweg dat "artificiële intelligentie niet-biologische intelligentie is".
Volgens Boden (2016) verwijst AI naar computers die de dingen doen die men met verstand doet. "Sommige van deze zaken (bv. redeneren) worden doorgaans omschreven als 'intelligent'. Andere (bv. zicht) niet. Maar allemaal vereisen ze psychologische vaardigheden die het voor een mens en een dier mogelijk maken om hun doelen te bereiken, zoals perceptie, associatie, voorspelling, planning en het controleren van een motor".

AI heeft inderdaad al succesvolle toepassingen: gezichtsherkenning in China om toegang te verlenen tot gebouwen, AI-systemen die specialisten evenaren in de diagnose van huidkanker, AlphaGo Zero dat zelf go leerde spelen enkel door te spelen tegen zichzelf (Steels et al., 2017).

Maar hoewel er ongetwijfeld spannende ontwikkelingen zijn op het vlak van AI, staat de kunstmatige intelligentie vooralsnog mijlenver af van menselijke intelligentie. Vandaag de dag is een AI-systeem nog steeds gericht op het invullen van een bepaalde taak en is maar goed in datgene waarvoor het ontworpen is.

AI is al aanwezig in het dagelijks leven: gezichtsherkenning op Facebook, Google Translate, vragen stellen aan Siri, de chatbot van

Men maakt een onderscheid tussen *general AI* en *narrow AI*. Bij ***general AI*** gaat het over computersystemen met dezelfde capaciteiten als een mens. We hebben vooralsnog niet de kennis om dergelijke systemen te ontwerpen. Voorlopig zit AI nog maar op het niveau van de ***narrow AI***: AI-systemen die de specifieke taken doen waarvoor ze ontworpen werden.

bol.com, suggesties op Netflix ...

AI is echter ook al doorgedrongen in de juridische wereld en de gezondheidszorg. Dat men met AI probeert om wiskundige bewijzen op te stellen, verbaast niemand, maar AI speelt ook een rol bij het opsporen van kunstvervalsingen en geeft nieuwe onderzoeksdomeinen zoals computerlinguïstiek een boost. Dus hoewel er jobs zullen verdwijnen door de opkomst van AI, zullen er ook nieuwe jobs door ontstaan.

Hoewel men optimistisch is over de toekomstige ontwikkelingen binnen AI, is **realiteitszin** op zijn plaats. Men moet beseffen dat sommige zaken veel tijd zullen vergen, bv. vertalen tussen om het even welke talen en het begrijpen van taal.

Opdracht: toepassingen van AI

- Welke toepassingen van AI ken je?
- Met welke toepassingen ben je zelf al in contact gekomen?
- Zoek op het internet, in kranten of tijdschriften een toepassing die je nog niet kende.

5.2 Geschiedenis van AI

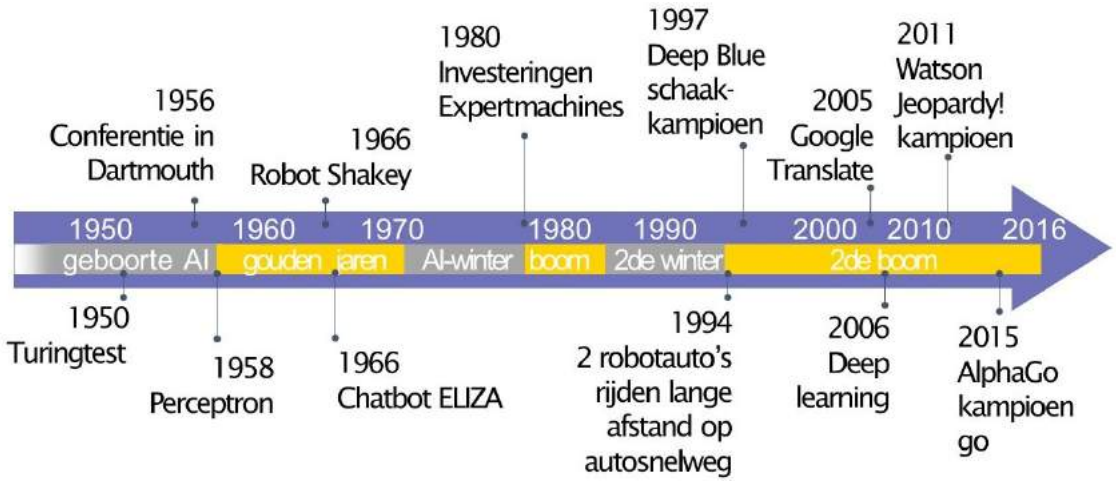
In de negentiende eeuw ontwierp Charles Babbage de 'Analytical Machine' en publiceerde **Ada Lovelace** het eerste computerprogramma. Bovendien dacht Lovelace toen al na over de mogelijkheden die zo'n machine zou hebben.

In 1950 vroeg **Alan Turing** zich af of een machine er in een conversatie zou kunnen in slagen een mens te doen geloven dat zij een mens is. Bij de Turingtest stelt een mens vragen aan een ongekende partij in een andere kamer, die zowel een mens als een machine kan zijn, om dan te besluiten of die een mens is of niet. Hiermee gaf Turing een eerste aanzet tot het onderzoeksdomein dat we nu kennen als kunstmatige intelligentie (KI of AI).

AI is dus geen nieuw fenomeen (zie Figuur 5.1). Het onderzoeksdomein bestaat al sinds **1956**, toen op een **conferentie in Dartmouth** vooraanstaande onderzoekers waaronder John McCarthy, Marvin Minsky, Claude Shannon en Nathaniel Rochester (zeer optimistisch) stelden dat "elk aspect van leren of elk ander kenmerk van intelligentie in principe zo precies kan worden beschreven dat er een machine gemaakt kan worden om het na te bootsen".

In 1958 ontwierp Frank Rosenplatt het **Perceptron**, een *neuraal netwerk* waarbij inputs direct met outputs zijn verbonden. De New York Times berichtte hierover met veel sensatie hoewel er bewezen was dat het Perceptron slechts lineair scheidbare patronen kon herkennen.

De robot **Shakey** uit 1966 werd gecreëerd aan het Artificial Intelligence Center van het Stanford Research Institute en was de eerste



Figuur 5.1: Geschiedenis AI. Afbeelding uit presentatie Accenture, met toestemming van L. Depuydt (persoonlijke communicatie).

robot die instructies kon analyseren en opbreken in deelinstructies. Het project combineerde *robotics*, *computer vision* en *natural language processing*.

Op het Massachusetts Institute of Technology (MIT) ontwierp Weizenbaum de software **ELIZA** waarmee een Rogeriaanse psychotherapeut bij een intakegesprek van een nieuwe patiënt werd nagespeeld. ELIZA kan beschouwd worden als de eerste chatbot (Güzeldere & Franchi, 1995).

Figuur 5.3: Conversatie met Eliza (Eliza, 2018).

Figuur 5.2: Robot Shakey (Nardone, 2007).

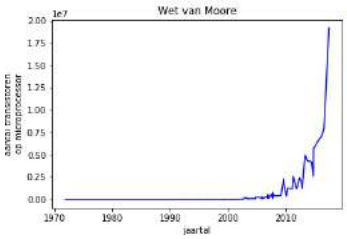
Tussen 1974 en 1980 kende het onderzoeksdomein een '**AI-winter**' waarbij de ontwikkelingen in AI een terugval kenden. De overheid en investeerders hadden veel kritiek op het onderzoek, maar desondanks boekten de onderzoekers toch vooruitgang. In die tijd speelde het gebrek aan rekenkracht de onderzoekers parten.

In de jaren 80 werden wel weer **investeringen** gedaan. Expertsystemen die de besluitvorming van menselijke experts om zeer specifieke problemen op te lossen simuleerden, zoals het diagnos-

De wet van Moore stelt dat elke twee jaar het aantal transistoren op een microchip verdubbelt. Daardoor ontstaat steeds meer rekenkracht.

ticeren van besmettelijke ziektes of het identificeren van chemische componenten, waren toen zeer populair.

Maar de desktopcomputers van Apple en IBM werden steeds sneller en krachtiger, waardoor de markt van de gespecialiseerde AI-hardware instortte. Er was sprake van een **tweede AI-winter** tussen 1987 en 1993 (Lim, 2018).



Figuur 5.4: Het aantal transistoren op een microcontroller. Data van Rupp (Rupp, 2019).

Sinds **midden jaren 90** is er een **opleving**. In 1994 reden voor het eerst twee zelfrijdende auto's op de autosnelweg in de buurt van Parijs, te midden van het drukke verkeer. In 1997 versloeg IBM Deep Blue de heersende schaakkampioen. In 2006 werd een online versie van Google Translate gelanceerd. Er kon vertaald worden van het Engels naar het Arabisch en vice versa. In 2011 werd IBM's Watson kampioen in de quiz Jeopardy!. Naast kennis was hierbij ook *natural language processing* belangrijk. In 2016 won Google DeepMind's AlphaGo zelfs van de wereldkampioen in go (zie Figuur 5.1).

De laatste jaren kon men dankzij **snellere hardware** en **zeer grote datasets** veel vooruitgang boeken in het machinaal leren (*machine learning*, ML).

Door de ontwikkelingen in de **game industrie** en in de **dataopslag**, maar vooral door de mogelijkheden die het **internet** biedt om zeer grote datasets te verzamelen en te delen, is ML prominent aanwezig sedert het begin van de jaren 2010.

Sommigen veronderstellen dat de snelle ontwikkelingen in AI zullen leiden tot het ontstaan van kunstmatige intelligente systemen die zichzelf zullen verbeteren zonder menselijke tussenkomst. Ze zullen zó intelligent worden dat ze de maatschappij kunnen sturen en zullen overnemen. De creatie van zo'n artificiële superintelligentie is gekend als de '**technologische singulariteit**'. De technologische singulariteit is een hypothese. Sommigen denken dat het nooit zal gebeuren, maar anderen verwachten dat deze singulariteit al in deze eeuw zal worden bereikt.

5.3 Soorten AI

Uit het voorgaande blijkt dat er meerdere soorten AI bestaan.

Om AI in te zetten voor een bepaald probleem, wordt informatie over dat probleem aan de computer gegeven. Vervolgens verwerkt het AI-systeem deze informatie en komt er een output. Men kan daarbij het probleem kennisgebaseerd of datagebaseerd aanpakken.

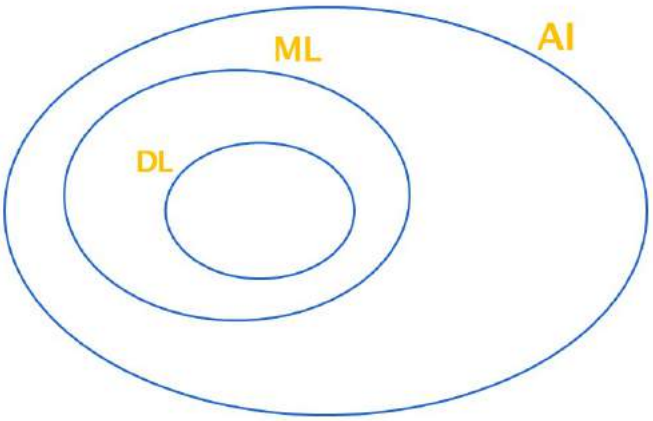
Een **kennisgebaseerde aanpak** houdt in dat men de kennis van menselijke experts zoveel mogelijk in regels probeert te gieten om de kennis van deze experts eigen te maken aan een expertsysteem.

Bij een **datagebaseerde** aanpak worden met statische methodes patronen in relevante data opgespoord en dan gebruikt om nieuwe

problemen op te lossen. Men spreekt dan van machinaal leren (*machine learning*, ML) (Steels et al., 2017).

Kennisgebaseerde systemen kwamen veelvuldig voor in de voorbeelden van paragraaf 5.2. Robot Shakey, ELIZA en Deep Blue zijn voorbeelden van zulke systemen. Deepmind's AlphaGo en Google Translate zijn daarentegen machine learning-systemen.

Binnen het machinaal leren bekleedt *deep learning* (DL) een prominente plaats (zie Figuur 5.5). DL heeft een revolutie veroorzaakt in het veld van ML door opmerkelijke resultaten te boeken op het vlak van spraak- en beeldherkenning, wat vaardigheden zijn waarvan de computer de menselijke prestaties tot op vandaag moeilijk kan evenaren.



Figuur 5.5: AI, ML en DL.

5.4 Nieuwe technologieën

Nieuwe technologieën veranderen de maatschappij: sommige doen dat op een zeer ingrijpende manier. Hoe ingrijpend een nieuwe technologie de maatschappij zal veranderen, is moeilijk in te schatten.

De impact van AI zal mogelijk nog groter zijn dan de komst van het internet.

Video

Mobiel bellen in 1998.
`https://youtu.be/TNwhIHqM60g` (Bromet, 2016).

Mobiel bellen: 1998 vs NU.
`https://youtu.be/mQVkpdzPGtQ` (DWDD, 2019).

5.5 Ethiek

Net zoals bij andere technologische ontwikkelingen steken ook hier ethische dilemma's de kop op. Bovendien kunnen er ook moreel ongewenste effecten optreden waar men bij de ontwikkeling van het AI-systeem niet op bedacht was.

Als men de AI-technologie goed inzet, kan die bijdragen tot een betere wereld. Denk maar aan exoskeletons die mensen met een beperking meer mobiel maken, ouderen die langer thuis kunnen blijven wonen, een meer efficiënte voedselproductie, zuinigere elektrische toestellen, veiliger werk door robots in te zetten bij de ontmijningsdienst.

Maar zal AI te allen tijde worden ingezet voor een mooiere maatschappij? Bij de ontwikkeling van AI-systemen kan men zich door verschillende motieven laten leiden. Zal men bij de ontwikkeling van een nieuw product steeds de mens centraal zetten, of soms ook het AI-systeem zelf of geldbejag?

In tegenstelling tot wat men doorgaans denkt, neemt een AI-systeem geen 100 % 'objectieve' beslissingen. Een AI-systeem is '**vooringenomen**', meestal te wijten aan de data waarmee het systeem werd getraind. Deze *bias* moet men zoveel mogelijk controleren en tot het minimum herleiden (zie kader 'Bias'). **Ontwikkelaars van AI-systemen moeten reeds bij de ontwikkeling bedacht zijn op mogelijke ongewenste effecten.** De **morele verantwoordelijkheid** van een AI-systeem dat in de maatschappij wordt gebracht, ligt niet bij het AI-systeem, maar bij de mens.

Voorbeeld: Tot 2018 gebruikte Amazon een sterk bejubeld maar ondertussen afgevoerd AI-systeem om sollicitanten te beoordelen. Het systeem selecteerde geen vrouwen voor technologische posities. Het was immers getraind met historische data: sollicitanten van de voorbije 10 jaar, voornamelijk mannen aangezien zij nog steeds de technologiewereld domineren.

Het nemen van beslissingen wordt steeds meer geautomatiseerd. Zo wordt er bepaald welk nieuws je te zien krijgt op Facebook, welke filmpjes op YouTube worden geweigerd, maar ook de banken en de overheid maken er gebruik van. Daarom klinkt de vraag naar transparantie steeds luider. Men moet er zich echter ook bewust van zijn dat transparantie over het gebruikte algoritme nog niet betekent dat het ook betrouwbaar is.

Ethische kwesties vergen veel tijd. Ze zijn ook vaak complex. Men mag daarom niet wachten tot de technologie klaar is om de nodige ethische discussies te voeren. Er is in elk geval al aandacht voor: *ethical AI*, *responsible AI*, *AI4good*, *trustworthy AI*, *explainable AI* zijn termen die her en der opduiken, met de bijbehorende richtlijnen.

Bias

Bias komt voor als de data niet representatief zijn. Als men bv. enkel groene appels als voorbeelden geeft aan een DL-systeem, dan zal het model geen rode appels herkennen. Of als men enkel foto's van honden onder een stralend blauwe hemel aanbiedt, dan zal het DL-model een hond in de regen niet bij de klasse 'hond' indelen.
Als de gebruikte data gekleurd zijn door een aanwezige bias in de maatschappij, zoals stereotypen, dan zal dit ook doorgegeven worden aan het ML-systeem. Men moet er dus over waken dat het model daardoor niet discriminerend wordt voor bepaalde bevolkingsgroepen. Als men bv. enkel vrouwelijke verplegers in de dataset stopt, dan zullen mannen niet als verpleger worden geclassificeerd.
Een model wordt nochtans getest voor het in gebruik genomen wordt. De testdata kunnen echter dezelfde bias bevatten als de trainingdata.
Het KIKS-model bevat ook een bias. De foto's van de trainingset zijn zo gemaakt dat de huidmondjes erop ongeveer passen in een vak van 120 op 120 pixels. Deze voorbeelden zijn afdrukken van bladeren genomen met transparante nagellak. Dat heeft als gevolg dat het model het best zal presteren op afbeeldingen van even grote huidmondjes in een gelijkaardige kleur.

Video

This 'Racist soap dispenser' at Facebook office does not work for black people.
`https://youtu.be/YJjv_OeiHmo` (Futureism, 2017).

Opdracht - bias in AI

- Zoek op het internet, in kranten of tijdschriften een geval van bias in een AI-systeem dat reeds gebruikt wordt (werd).

Samengevat

De term ’artificiële intelligentie (AI)’ bestaat reeds sinds de jaren 50 en de ontwikkeling ervan kende ups en downs. De mindere periodes zijn gekend als AI-winters.
Van AI doen veel definities de ronde. AI behelst immers zoveel verschillende aspecten en invalshoeken dat een algemene definitie moeilijk is.

Binnen de AI-systemen onderscheidt men kennisgebaseerde en datagebaseerde systemen.

Machine learning (ML) omvat *deep learning* (DL), dat een revolutie veroorzaakt heeft in het veld van ML door opmerkelijke resultaten te boeken op het vlak van spraak- en beeldherkenning.

Nieuwe technologieën veranderen de maatschappij en sommige doen dat op een zeer ingrijpende manier. De impact van AI zal mogelijk nog groter zijn dan de komst van het internet.
AI is al aanwezig in het dagelijks leven en is ook al doorgedrongen in domeinen zoals de juridische wereld, de gezondheidszorg, de kunstwereld en de computerlinguïstiek.
Hoewel men optimistisch is over de toekomstige ontwikkelingen binnen AI, is enige realiteitszin op zijn plaats.
Net zoals bij andere technologische ontwikkelingen steken ook hier ethische dilemma’s de kop op. Een AI-systeem is ‘vooringenomen’. Deze *bias* moet men zoveel mogelijk controleren en tot het minimum herleiden.

Na dit hoofdstuk weet je dat er verschillende soorten AI bestaan. Je weet dat machine learning en deep learning datagebaseerde AI-systemen zijn.
Je kent het verschil tussen narrow AI en general AI.
Je weet wat bias in een AI-systeem is.
Je weet wat een AI-winter is.

HOOFDSTUK
ZES

DIGITALE BEELDEN

6.1 Pixel

Notebooks

- Doorloop de notebook `MatricesAfbeeldingenGrijswaarden.ipynb` in de map `IntroductiePython`.

Als je genoeg inzoomt op een bepaald deel van de foto, dan kan je de pixels onderscheiden zoals in Figuur 6.1.

Figuur 6.1: Inzoomen op foto.

Digitale afbeelding

Een **digitale afbeelding** is een rechthoekig rooster van pixels.

Men spreekt ook van een rasterafbeelding of een bitmap. GIF, JPEG en PNG zijn bestandsformaten voor het opslaan van rasterafbeeldingen in digitale vorm.

Elke **pixel heeft een bepaalde kleur**:

- bij een zwart-witafbeelding is een pixel zwart of wit;
- bij een grijswaardenafbeelding is dat een grijswaarde;
- bij een kleurenafbeelding is dat een kleur.

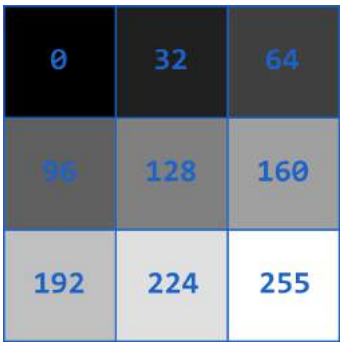
6.2 Grijswaarden

Bekijk de foto in Figuur 6.2. Deze foto is in grijswaarden en de pixels zijn zichtbaar.

Figuur 6.2: Foto in grijswaarden.

Een grijswaarde wordt door de computer voorgesteld door een natuurlijk getal tussen 0 en 255. Hierbij komt 0 overeen met zwart en 255 met wit.

De waarden ertussen geven dus grijswaarden weer die stilaan lichter worden naarmate het getal stijgt. Dit wordt geïllustreerd in Figuur 6.3.



Figuur 6.3: Elk getal van 0 t.e.m. 255 komt overeen met een bepaalde grijswaarde.

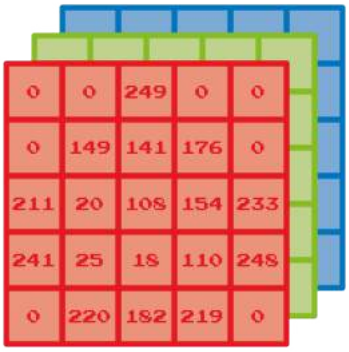
Dit betekent dat je een **afbeelding in grijswaarden** kan voorstellen met een **rechthoekig rooster van getallen tussen 0 en 255**.

In de wiskunde spreekt men van een **matrix** waarvan de elementen natuurlijke getallen tussen 0 en 255 zijn.

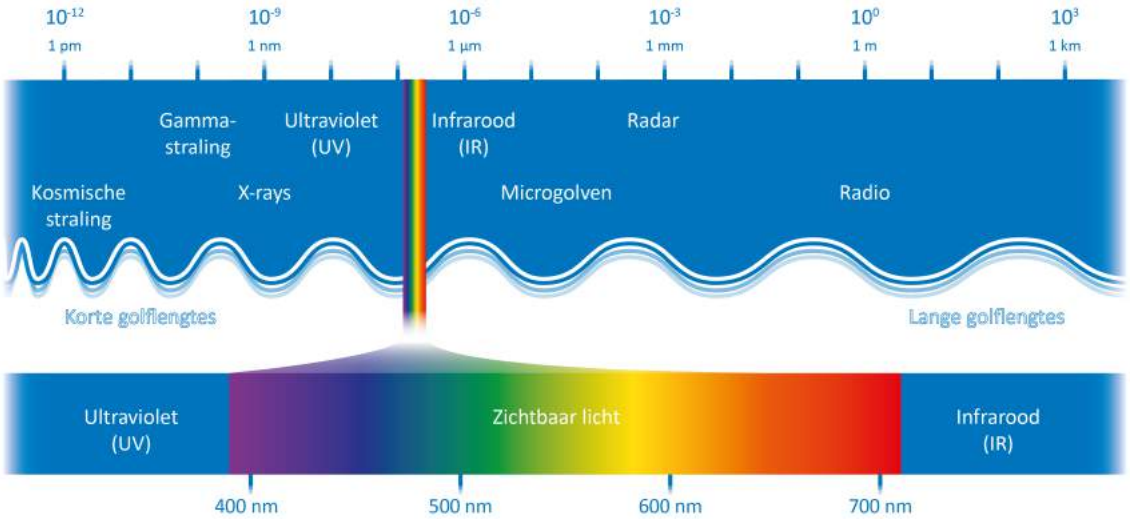
6.3 Kleuren

Elektromagnetische golven hebben een breed spectrum van golflengtes, waarbij elke **golflengte** overeenkomt met een andere **kleur** (zie Figuur 6.5). Het licht dat de mens kan zien, zichtbaar licht, beslaat slechts een klein deel van het spectrum. Het zichtbaar licht met de kleinste golflengtes is blauw, dat met de grootste golflengtes is rood.

Door rood, groen en blauw licht samen te brengen, kan bijna elke kleur gesimuleerd worden. Kleurenfoto's kunnen dan ook in een RGB-systeem worden opgeslagen (RGB = rood, groen, blauw). In dat geval zijn er wel **drie matrices** nodig: één voor de rode tinten, één voor de groene tinten en één voor de blauwe tinten. Elk van die matrices bevat getallen tussen 0 en 255, waarbij de waarde van het element de intensiteit van de tint weergeeft. **Deze drie roosters worden dan samengevoegd tot een soort balk van getallen.** In de computerwetenschappen spreekt men van een '**tensor**'



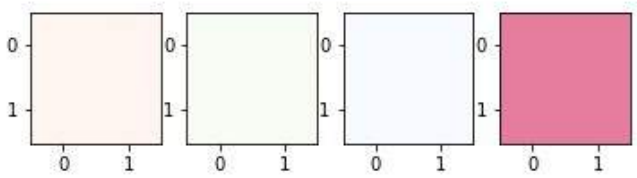
Figuur 6.4: Tensor van een kleurenafbeelding.



Figuur 6.5: Elektromagnetische golven.

(zie Figuur 6.4).

In Figuur 6.6 worden de beelden van drie gekozen matrices en de overeenkomstige tensor getoond. De rode tint heeft waarde 230, de groene tint 125 en de blauwe tint 156. De kleur van het laatste vierkant heeft dus als RGB-code 230, 125, 156.



Figuur 6.6: Drie matrices vormen een tensor die een kleurenafbeelding representeert.

Notebook (facultatief)

- Wil je meer in detail weten hoe het zit met kleurenfoto's, dan kan je de notebook `TensorenRgb.ipynb` onder de loep nemen.
- Meer uitleg over tensoren vind je in de notebook `Tensoren.ipynb`.

6.4 Type van de elementen van een matrix of een tensor in Python

Een matrix en een tensor worden in Python voorgesteld door een NumPy **ndarray**. De matrices en de tensoren bevatten getallen tussen 0 en 255 die een grijswaarde of een intensiteit van een tint weergeven.

In `NumPy` kan men ervoor kiezen dat deze elementen het type `uint8` (*8 bits-unsigned integer*) hebben. Voor de opslag van een element wordt dan gebruikgemaakt van 8 bits, m.a.w. een byte

(zie kader 'Bit en byte'). De natuurlijke getallen van 0 t.e.m. 255 voorgesteld in het tiendelige talstelsel, worden dan gerepresenteerd in het binaire talstelsel (zie kader 'Binaire talstelsel').

Notebook

- Het type `uint8` wordt behandeld in de notebook `DatastructuurNumPy.ipynb` in de map `IntroductiePython`.

Bit en byte

Een 'bit' is een informatie-eenheid. De term is afkomstig van *binary digit*. Het is een eenheid die enkel de waarden 0 en 1 kan aannemen. Acht bits vormen samen een 'byte'. Er zijn 256 mogelijke combinaties mogelijk van 0 en 1 die samen een byte vormen.

Binair talstelsel

In het tiendelige talstelsel worden de getallen voorgesteld m.b.v. de cijfers 0 t.e.m. 9. Het binaire talstelsel gebruikt daarvoor enkel de cijfers 0 en 1.
Zoals in het tiendelige talstelsel het getal $123 = 3.10^0 + 2.10^1 + 1.10^2$, is in het binaire talstelsel het getal $101 = 1.2^0 + 0.2^1 + 1.2^2$, dus 5. Met elk cijfer in een getal komt dus als waarde een veelvoud van een macht van resp. 10 en 2 overeen. Het veelvoud wordt bepaald door het cijfer, de exponent van de macht door de plaats van het cijfer in het getal.
Het binaire talstelsel heeft grondtal 2, het decimale talstelsel 10.

Soms worden echter ook waarden tussen 0 en 1 gebruikt i.p.v. tussen 0 en 255. Je kan dat bekomen door de data te **normaliseren**, m.a.w. alle waarden te delen door 255. De computer werkt dan **verhoudingsgewijs** voor de tinten, waarbij 0 overeenkomt met 0 en 1 met 255. Dat betekent ook dat de elementen dan een ander type hebben, het type `float`.

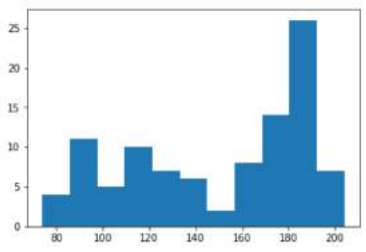
Notebook - formaat (facultatief)

- Wil je een grijswaardenafbeelding maken van een kleurenafbeelding en je afbeelding ook zo bewaren? Bekijk dan de notebook `VanJPGnaarDATofNPY.ipynb` in de map `IntroductiePython` om te zien hoe je dat kunt doen.

6.5 Verdeling van de kleuren

In Python kan je een verdeling van de kleuren of van de intensiteit van een tint opvragen in de vorm van een **histogram**.

Je kan daarbij van elk getal tussen 0 en 255 laten weergeven hoeveel keer het voorkomt of je kan het interval [0,255] verdelen in deelintervallen en van elk deelinterval de frequentie visualiseren zoals in Figuur 6.7.



Figuur 6.7: Verdeling van grijswaarden.

Notebook

- Deze histogrammen komen aan bod in de notebook `MatricesAfbeeldingenGrijswaarden.ipynb` in de map `IntroductiePython`.

6.6 Toepassing: verborgen boodschap (facultatief)

Door op een specifieke manier te 'spelen' met de waarden in de matrix of tensor die een afbeelding representeert, kan je er een boodschap in verbergen.

Notebook - verborgen boodschap (facultatief)

- Ga op zoek naar de verborgen boodschap in de notebook `VerborgenBoodschap.ipynb` in de map `IntroductiePython`.

Een pixelwaarde met één verminderen of vermeerderen zal geen zichtbaar effect hebben op de afbeelding, maar je kan ermee wel het merendeel van de pixels bv. even maken. Als je ervoor zorgt dat de pixels waar de boodschap zich bevindt oneven zijn, dan kan je de boodschap onthullen met enkele lijntjes code.

Samengevat

Een digitale afbeelding is een rechthoekig rooster van pixels. Elke pixel heeft een bepaalde kleur: bij een zwart-witafbeelding is een pixel zwart of wit, bij een grijswaardenafbeelding is dat een grijswaarde en bij een kleurenafbeelding is dat een kleur.

Grijswaarden worden door de computer voorgesteld door een natuurlijk getal tussen 0 en 255. Hierbij komt 0 overeen met zwart en 255 met wit. De waarden ertussen geven dus grijswaarden weer die stilaan lichter worden naarmate het getal stijgt. Een afbeelding in grijswaarden kan men voorstellen met een matrix waarvan de elementen natuurlijke getallen tussen 0 en 255 zijn.

Kleurenfoto's worden in een RGB-systeem opgeslagen. In dat geval zijn er drie matrices nodig die samen een tensor vormen. Deze matrices geven de intensiteit weer van respectievelijk de rode, groene en blauwe tinten aanwezig in de foto.

Samengevat - computerwetenschappen

Je weet hoe een computer naar een afbeelding kijkt.
Je weet wat een matrix is. Je weet wat een tensor is.

Je weet wat de volgende begrippen betekenen: pixel, RGB, grijswaarden, bit, byte, normaliseren.

Je kunt de pyplot-functies `imshow()` en `hist()` en de NumPy-functies `min()`, `max()`, `load()`, `ravel()` en `zeros()` gebruiken. In pyplot werkte je met de parameter `bins`.

Je weet hoe je in Python een stuk uit een foto kunt halen.
Facultatief werkte je met kleurenafbeeldingen in Python. Dan weet je hoe je in Python de rode, groene en blauwe tinten in een kleurenafbeelding van elkaar kunt scheiden en gebruikte je de NumPy-functie `dstack()` en `imread()`.

HOOFDSTUK
ZEVEN

FUNDAMENTEN VAN MACHINAAL LEREN

7.1 Machine learning

7.1.1 Machine learning-algoritmes

Machine learning (ML) is een populair en succesvol onderdeel van de artificiële intelligentie. Machine learning is een computerwetenschappelijke discipline waarin men vooral proefondervindelijk te werk gaat, maar die wel **wiskundig** onderbouwd is, en waarin men gebruikmaakt van principes uit de wiskundige **statistiek** (Chollet, 2018).

Een machine learning-systeem verwerft met lerende algoritmes kennis uit data met de bedoeling uitkomsten te kunnen voorspellen betreffende nieuwe data. Deze voorspellingen worden gedaan met een bepaalde zekerheid. Een ML-systeem neemt zijn beslissingen dus niet op basis van vooraf in detail geprogrammeerde instructies.

De nieuwe data moeten wel gelijkaardig zijn aan de aangeboden data.

Voorspellen betekent bijvoorbeeld dat er uit voorbije tendensen cijfers voor de toekomst gegenereerd worden of dat een object bij een bepaalde klasse wordt ingedeeld.

Lerende algoritmes zijn algoritmes waarbij het ML-systeem zelf gaandeweg aanpassingen doet aan de aanwezige parameters tijdens het leerproces, om geleidelijk aan te komen tot betere prestaties.

Uitleg over deze parameters die worden aangepast, vindt u in hoofdstuk 10.

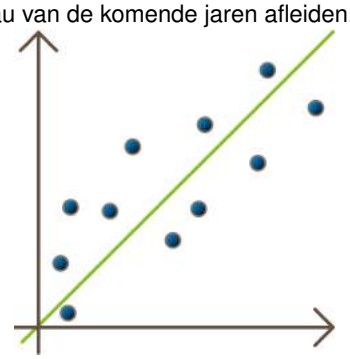
Met ML kan je bv. problemen van classificatie en van regressie aanpakken (zie kader 'Regressie en classificatie').

Notebook - classificatie en regressie

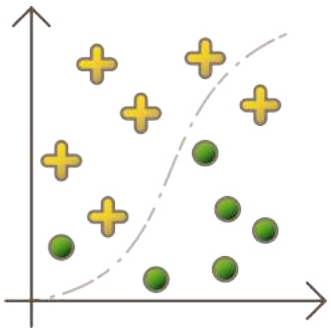
- Misschien bekeek je reeds een notebook over regressie: `Zeeniveau.ipynb` in de map `IntroductieMachineLearning`.
- In dezelfde map worden er ook regressieproblemen behandeld in de notebooks over de Morteratsch- en Silvrettagletsjer, `MorteratschgletsjerRegressie.ipynb` en `SilvrettagletsjerRegressie.ipynb`, en in de notebooks `StomataHoogtebomen.ipynb` en `IrisviriginicaRegressie.ipynb`.
- In het volgende hoofdstuk komen er notebooks over classificatie aan bod.

Regressie en classificatie

Bij regressie is het de bedoeling dat het ML-model een kromme of een oppervlak bepaalt dat het best bij de gegeven punten past. Eens deze kromme of dit oppervlak gekend is, kan het model voorspellingen doen naar uitkomsten bij nieuwe data. Bv. uit het zeeniveau van de voorbije decennia in Oostende, het zeeniveau van de komende jaren afleiden.



Figuur 7.1: Regressie.



Figuur 7.2: Classificatie.

Bij classificatie zijn er verschillende klassen waartoe de data kunnen behoren. Het is de bedoeling dat het ML-model van nieuwe data kan bepalen tot welke klasse ze behoren. Bv. van een foto kunnen zeggen of er al dan niet een huidmondje op staat.

7.1.2 Generaliseren

Het doel van ML is om modellen te bekomen die goed kunnen generaliseren, m.a.w. modellen die goed presteren op ongeziene data. Om dit te bewerkstelligen zijn er verschillende principes die je in acht kan nemen. **Je splitst bv. de dataset op in een *trainingset*, *valideringsset* en *testset*.**

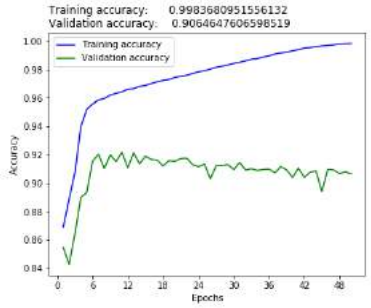
Het netwerk wordt getraind met de trainingdata en wordt tezelfdertijd geëvalueerd met de valideringsdata. Hierbij is het belangrijk dat de trainingset en de valideringsset disjunct zijn: ze bevatten dus geen gemeenschappelijke voorbeelden.

De trainingdata en valideringsdata worden meerdere keren doorlopen, men spreekt van meerdere epochs. Na elke *epoch* worden de prestaties op de trainingdata nagegaan. Op basis van deze informatie past het algoritme de aanwezige parameters na elke epoch aan.

Eens getraind, wordt het model één keer getest met de testdata.

De generalisatie vormt een uitdaging (zie Figuren 7.4, 7.5 en 7.6). Bij het begin van de training kiest het algoritme willekeurige waarden voor de parameters van het model. Het is dus niet verwonderlijk dat het netwerk initieel niet goed presteert. Na elke epoch past het algoritme de waarden van de parameters aan en verbeteren de resultaten. Zowel op de trainingdata als op de valideringsdata presteert het model steeds beter.

Zolang het netwerk echter nog niet alle relevante patronen in de trainingdata heeft ontdekt, zegt men dat het netwerk ***underfits***. Men



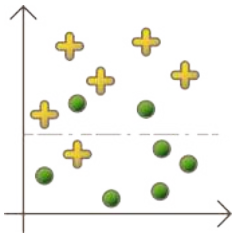
Figuur 7.3: Evolutie van de ***accuracy*** tijdens de training van een netwerk. De *accuracy* is het percentage correct geclassificeerde data. Het model overfits.

gaat dan door met het trainen van het netwerk. Het netwerk leert bij elke epoch meer uit de trainingdata. Na enige iteraties verbetert de generalisatie echter niet meer: het model presteert wel steeds beter op de trainingdata, maar daarentegen worden de resultaten op de valideringsdata slechter (zie Figuur 7.3). Dat komt omdat het model patronen begint te zien in de trainingdata die irrelevant zijn voor ongeziene data. Het model ***overfits***.

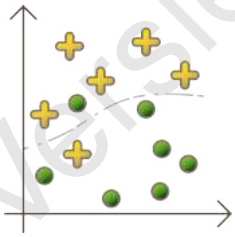
Het komt erop aan om de architectuur van het netwerk zo eenvoudig mogelijk te kiezen en op het juiste moment te stoppen met trainen om een zo goed mogelijk presterend model te bekomen.

Notebook - underfitting en overfitting (facultatief)

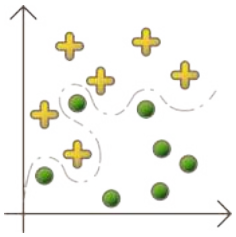
- Ook bij een regressieprobleem moet het model goed gekozen worden. Als je bv. punten gelegen op een parabool wilt benaderen door een best passende rechte, i.p.v. een parabool, dan zal je model blijven underfitten, ongeacht de duur van de training. Dit wordt gedemonstreerd in de notebook `Zeeniveau.ipynb` in de map `IntroductieMachineLearning`.
- In de notebook `Overfitting.ipynb` in de map `IntroductieDeepLearning` wordt dieper ingegaan op overfitting.



Figuur 7.4: Underfitting.



Figuur 7.5: Optimaal.



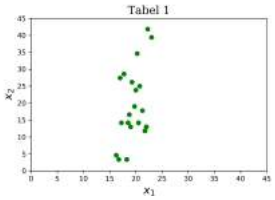
Figuur 7.6: Overfitting.

7.2 Standaardiseren en normaliseren van de data

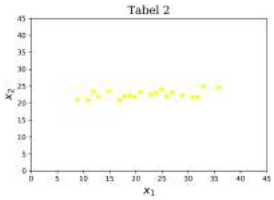
Notebook

- Doorloop de notebook `Standaardiseren.ipynb` in de map `IntroductieMachineLearning` en ontdek het belang van standaardiseren, als je gegevens van bv. verschillende grootteorde of in een verschillende eenheid met elkaar wilt vergelijken of in verband wilt brengen.

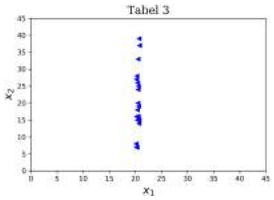
Om na te gaan hoe sterk de samenhang tussen **bivariate variabelen** is, zal je vaak de data visualiseren. De **correlatie** tussen bivariate gegevens kan je visueel proberen in te schatten door de overeenkomstige puntenwolk te bekijken.



Figuur 7.7: Tabel 1.



Figuur 7.8: Tabel 2.



Figuur 7.9: Tabel 3.

Op het eerste gezicht lijkt het dat de data in de derde puntenwolk de meeste samenhang vertonen.

Door de **correlatiecoëfficiënt** te berekenen, krijg je extra informatie. De correlatiecoëfficiënt r is een getal in [-1, 1] dat een maat is voor de samenhang tussen de bivariate gegevens. Hoe dichter $|r|$ bij 1 ligt, hoe meer samenhang er is.

De correlatiecoëfficiënt is voor alle drie de puntenwolken dezelfde, ongeveer $0{,}5$, waaruit volgt dat de samenhang van de gegevens voor elke puntenwolk dezelfde is.

De grootteorde van de veranderlijken en het bereik op de assen bepalen voor een groot deel hoe de grafiek die de gegevens representeert er zal gaan uitzien. Door de data te standaardiseren kan je dit effect neutraliseren. **Dus als je de correlatie van meerdere datasets op een grafiek wilt bekijken, dan werk je het best met gestandaardiseerde data. De vorm van de puntenwolk bij bivariate gegevens bv. is maar betrouwbaar als de gegevens gestandaardiseerd zijn.** Bekijk de puntenwolken van de gestandaardiseerde gegevens in Figuren 7.10, 7.11 en 7.12.

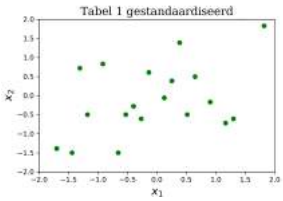
Het standaardiseren heeft trouwens geen effect op de waarde van de correlatiecoëfficient.

Je brengt de data in gestandaardiseerde vorm met de volgende **formule**:

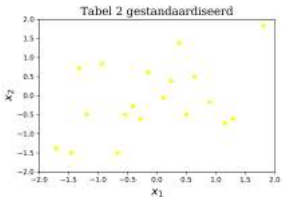
$$z = \frac{x - \bar{x}}{s(x)},$$

met $\bar{x}$ het gemiddelde en $s(x)$ de standaardafwijking van de x-waarden, d.i. de gemiddelde kwadratische afwijking t.o.v. het gemiddelde $\bar{x}$.

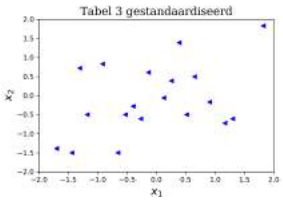
In de statistiek spreekt men van de Z-score.



Figuur 7.10: Tabel 1.



Figuur 7.11: Tabel 2.



Figuur 7.12: Tabel 3.

Bijkomende voordelen:

- Na het standaardiseren liggen de meeste waarden tussen 0 en 1, wat voordelig is voor het rekenen door de computer. Rekenen met vrij grote getallen leidt al snel tot nog grotere getallen en tot numerieke instabiliteit, dat is een bijkomende reden waarom de data worden gestandaardiseerd.

- Sommige algoritmes uit machinaal leren zijn pas bruikbaar als de data gestandaardiseerd zijn, omdat die algoritmes zo opgesteld zijn.

Vooral bij regressieproblemen zal men standaardiseren, omdat de correlatie daar tekenend is voor het probleem.

Bij classificatieproblemen kiest men er vaker voor enkel te normaliseren. Dan liggen alle waarden tussen 0 en 1.

Je brengt de data in genormaliseerde vorm met de volgende **formule**:

$$x_n = \frac{x}{x_{max} - x_{min}},$$

met x_{max} de grootste en x_{min} de kleinste x-waarde.

Bij het standaardiseren van gegevens komt het er dus op neer dat de waarden zo herschaald worden dat ze een gemiddelde 0 krijgen en een standaardafwijking 1.

Bij normalisering komt het erop neer dat de data herschaald worden naar waarden tussen 0 en 1.

7.3 Deep learning

Binnen het machinaal leren onderscheidt men *deep learning* (DL). Deep learning-modellen, m.a.w. **diepe neurale netwerken**, worden ingezet voor spraakherkenning en objectdetectie, maar ook voor het begrijpen van taal en toepassingen binnen de genetica. Google Translate en gezichtsherkenning op Facebook zijn voorbeelden van *deep learning*.

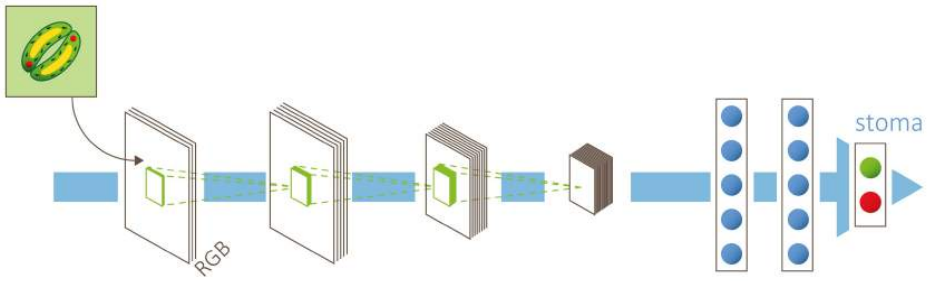
Hoewel men spreekt over neurale netwerken, hebben ze niet als doel om het menselijk brein te modelleren.

Een neuraal netwerk wordt opgebouwd door verschillende lagen (*layers*) aan elkaar te schakelen. Hoe meer lagen er zijn, hoe dieper het netwerk is. Men spreekt van een netwerk omdat er veel verschillende functies worden samengesteld.

Een neuraal netwerk is dus opgebouwd uit lagen: een invoer- en een uitvoerlaag (*input* en *output*), met daartussen de verborgen lagen (*hidden layers*) waarvan de outputs niet worden getoond. Elke laag bevat eenheden (*units*), ook neuronen of knopen genoemd. In elke laag werkt er een lineaire functie in op de invoer van die laag. Deze lineaire functie wordt bepaald door meerdere parameters, haar coëfficiënten, en wordt gevolgd door een niet-lineaire activatiefunctie. Zo bekomt men de uitvoer van die laag die dan dient als invoer van de volgende laag. Zo gaat de informatie 'voorwaarts', laag na laag; men spreekt van een ***feedforward*** netwerk.

Neuraal netwerk van KIKS

Bij het project KIKS werkt men met een convolutioneel neuraal netwerk.
Het AI-systeem van KIKS bv. leert relevante kenmerken onderscheiden omdat het vele voorbeelden van delen van bladeren te zien krijgt, voorbeelden met het label ‘stoma’ ofwel met het label ‘geen stoma’.
Als input krijgt het model een microfoto van een deel van een blad van een plant. Door o.a. randen te detecteren en die dan in volgende lagen te combineren tot een ovaal of tot de opening van een stoma, kan het netwerk uiteindelijk de stomata op het blad detecteren.



Figuur 7.13: Neuraal netwerk van KIKS.

Figuur 7.14: Het netwerk ontdekt geleidelijk aan meer patronen.

Een **convolutioneel neuraal netwerk** (convnet) is een diep neuraal netwerk dat zeer geschikt is voor beeldherkenning. Een convnet behoort tot het domein van supervised learning. Een convnet ontdekt in een eerste laag kleine lokale patronen zoals randen, in een tweede laag ziet het kenmerken die opgebouwd worden met de kenmerken uit de eerste layer. Daardoor kunnen deze netwerken op een efficiënte manier stijgende complexiteit en abstracte visuele concepten aan: randen vormen motieven, motieven vormen samen onderdelen en onderdelen vormen objecten (zie Figuur 7.14) (LeCun et al., 2015).

Samengevat - computerwetenschappen

Een machine learning-systeem verwerft met lerende algoritmes kennis uit data met de bedoeling uitkomsten te kunnen voorspellen betreffende nieuwe data, m.a.w. het systeem moet goed kunnen generaliseren.
Binnen ML onderscheidt men verschillende types van leren: *supervised*, *unsupervised*, en *reinforcement learning*.
Met ML kan men problemen van classificatie en regressie behandelen. Bij regressieproblemen zullen de data doorgaans eerst worden gestandaardiseerd, bij classificatieproblemen eerder genormaliseerd.

Om een ML-model te bekomen dat goed kan generaliseren, zijn er verschillende principes die men in acht kan nemen, zoals het opsplitsen van de dataset in een trainingset, valideringsset en testset.
Men moet waakzaam zijn voor *underfitting* en *overfitting*.

Binnen het machinaal leren onderscheidt men de diepe neurale netwerken die bijvoorbeeld worden ingezet voor beeldherkenning. Hoewel men spreekt over neurale netwerken, hebben ze niet als doel om het menselijk brein te modelleren.
Een diep neuraal netwerk wordt opgebouwd door verschillende lagen aan elkaar te schakelen: een invoer- en een uitvoerlaag, met daartussen de verborgen lagen. Elke laag bevat eenheden (neuronen) en parameters. Het trainen van het netwerk bestaat erin dat het algoritme van het netwerk de parameters op zo'n manier aanpast dat de gewenste output wordt bekomen.

Na dit hoofdstuk weet je wat lerende algoritmes zijn.
Je weet welke rol de learning rate speelt.
Je herkent underfitting en overfitting.
Je weet wat een diep neuraal netwerk is. Je maakte kennis met een convolutioneel neuraal netwerk.
Je kent de begrippen: lagen (layers), neuronen, gewichten (weights), activatiefunctie, trainen, standaardiseren en normaliseren.

HOOFDSTUK
ACHT

CONVOLUTIES

8.1 Een beeld filteren

Image filtering gebruik je bv. om ruis uit een foto te verwijderen, om het contrast te verscherpen of om randen te detecteren.

Convoluties zijn een bepaalde soort van deze filters. Voorbeelden zijn de gemiddelde-filter, de Sobelfilter en de Gaussfilter. Convoluties worden toegepast in de zogenaamde convolutionele neurale netwerken, zoals het KIKS-netwerk.

Voor convolutionele neurale netwerken: zie hoofdstuk 7.

Notebook

- Bekijk het effect van een convolutie in de notebook `Convolutie.ipynb` in de map `IntroductieDeepLearning`.

8.2 Convolutie

Met convoluties kan je dus op zoek gaan naar verschillende kenmerken in een afbeelding. Je kan er bv. verticale en horizontale lijnen mee detecteren, of het contrast in een beeld verzachten.

Convolutie

De convolutie is een wiskundige bewerking die enkel gebruikmaakt van optellen en vermenigvuldigen. Het komt erop neer dat men aan een pixel een bepaald gewicht geeft en men daaraan gewogen waarden van de omliggende pixels toevoegt.

8.3 De convolutiebewerking

Convoluties werken door een lineaire combinatie te nemen van de pixelwaarden van een pixel en zijn buren.

Notebook

- Hoe de convolutiebewerking werkt, leer je in de notebook `ConvolutieBewerking.ipynb`.

Bij een 'convolutie' laat men een 'filter' over een afbeelding 'glijden'; er wordt daarbij telkens één pixel opgeschoven. Zowel de afbeelding als de filter zijn matrices of tensoren. De elementen van de filter en de elementen van de matrix van de afbeelding worden elementsgewijs vermenigvuldigd en erna worden deze producten opgeteld.

Voorbeeld uit de notebook `ConvolutieBewerking.ipynb`: hoe verloopt dit concreet met een grijswaardenafbeelding?

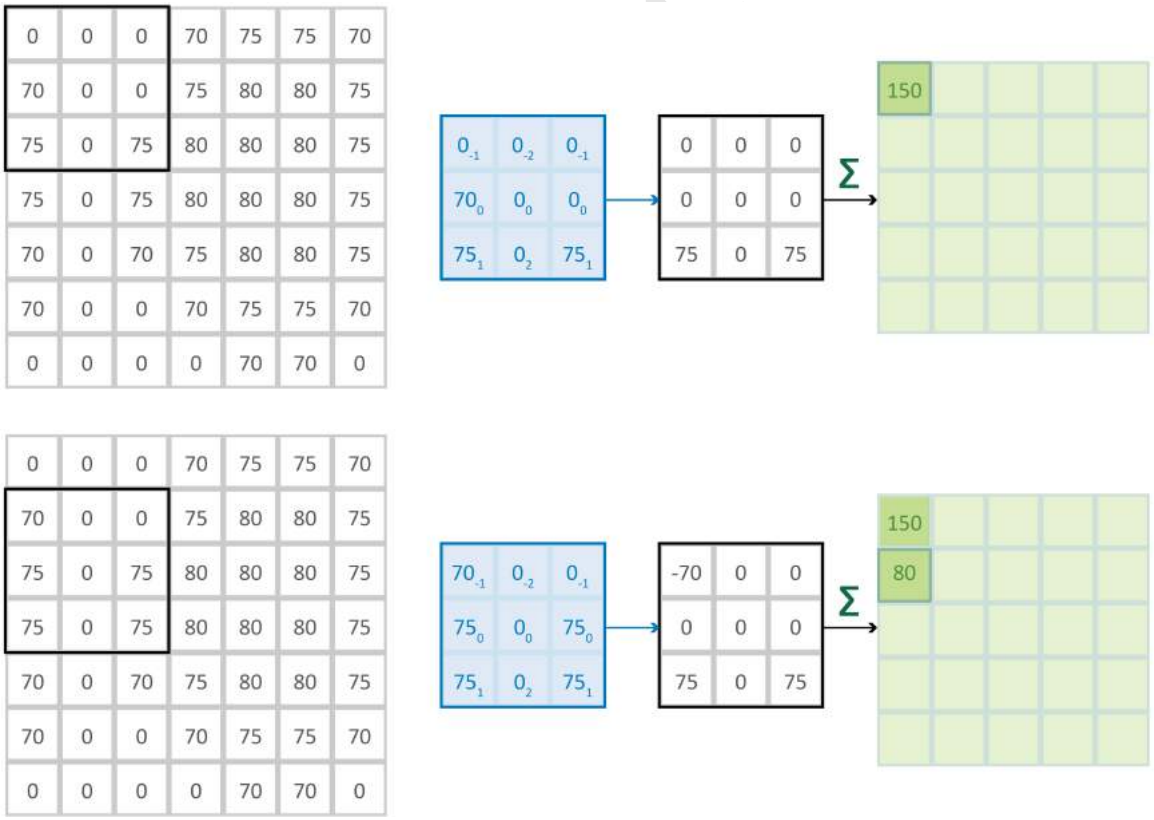
Beschouw de Sobelfilter.

Deze filter is een $3x3$-matrix: $\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$.

Eerst 'flip' je de filter: je manipuleert de matrix door de eerste en de derde rij te verwisselen en vervolgens hetzelfde te doen met de eerste en de derde kolom.

Je bekomt de matrix: $\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$.

Deze matrix laat je over de afbeelding glijden. Het rekenwerk dat gebeurt, wordt geïllustreerd in Figuur 8.1.



Figuur 8.1: Convolutiebewerking. Gebaseerd op afbeelding van Rob Robinson (Robinson, 2017).

Er rest niets anders dan de overeenkomstige elementen van de matrix en de filter op elke positie met elkaar te vermenigvuldigen en erna alle producten op te tellen.

Bv. in de eerste positie: $0 \cdot (-1) + 0 \cdot (-2) + 0 \cdot (-1) + 70 \cdot 0 +$

$0 \cdot 0 + 0 \cdot 0 + 75 \cdot 1 + 0 \cdot 2 + 75 \cdot 1 = 150.$

Opdracht - convolutie met de hand

- Pas zelf de convolutiebewerking toe en vul de groene matrix in Figuur 8.1 aan.

Het resultaat van de convolutie is een matrix die kleiner is dan de oorspronkelijke matrix. Men 'verliest' twee pixels in elke dimensie. In het voorbeeld in Figuur 8.1 gaat de $7x7$-matrix over in een $5x5$-matrix.

8.4 Convolutie uitvoeren in Python

In de notebook 'Convolutie' pas je een filter toe op een grijswaardenfoto van bamboe om verticale lijnen te detecteren.

Om verticale lijnen te detecteren, gebruik je er de filter $\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$.

Je geeft deze filter in in Python als een NumPy-array. Om de convolutie uit te voeren, gebruik je de functie `convolve2d()`. Deze functie is voorhanden in de module signal voor beeldverwerking van de Python-module SciPy. Merk op dat het 'flippen' van de matrix reeds inbegrepen zit in de functie `convolve2d()`.

Concrete werkwijze:

- Importeer de modules `NumPy`, `scipy.signal` en `matplotlib.pyplot`.
- Voer de filter in.
- Voer een foto in; de foto is een grijswaardenafbeelding.

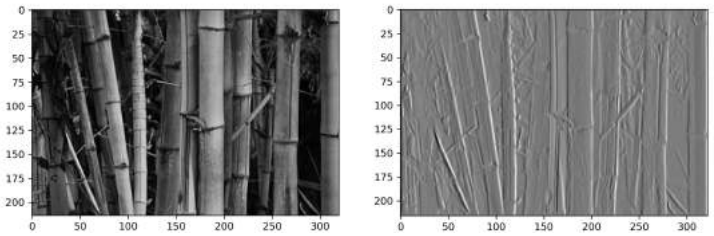
```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> import scipy.signal

>>> vertic_filter = np.array([[-1,0,1],
                              [-1,0,1],
                              [-1,0,1]])
>>> bamboe =  np.load("bamboe.npy")
```

- Voer de convolutie uit:

```
>>> bamboe_vertic = scipy.signal.convolve2d(bamboe,
               vertic_filter,
               mode="valid")
```

Het resultaat zie je in Figuur 8.2.



Figuur 8.2: Originele en gefilterde foto. De linkse foto is een afgeleide van het werk van McGrath (2007).

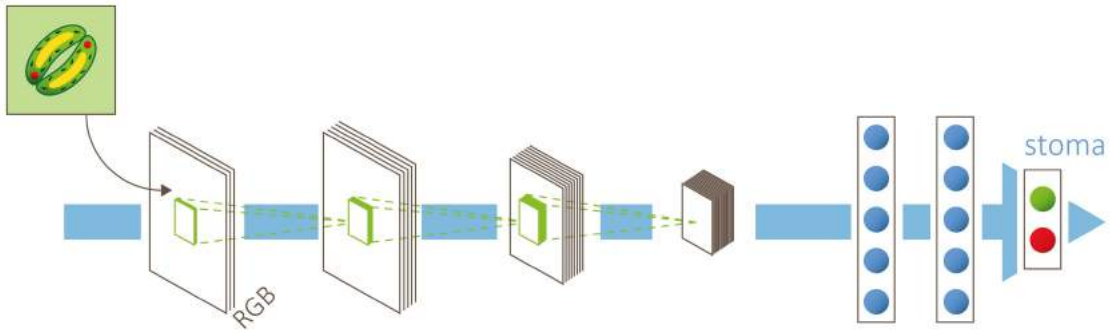
8.5 *Convolutioneel neuraal netwerk van KIKS*

Om huidmondjes te herkennen, gebruikt KIKS een diep neuraal netwerk gevolgd door een netwerk met ***dense layers*** dat bestaat uit een convolutioneel netwerk, (zie Figuur 8.3).

Dense layers: de neuronen in een bepaalde laag zijn verbonden met alle neuronen in de vorige laag.

In één laag van een convolutioneel netwerk gebeuren er meerdere convolutiebewerkingen. Bv. in een laag met 24 neuronen worden er 24 convolutiebewerkingen gedaan. Elke convolutie levert een matrix op. Op deze matrix wordt vervolgens een niet-lineaire activatiefunctie toegepast, wat opnieuw een matrix oplevert.

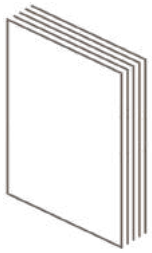
Voor de niet-lineaire activatiefunctie ReLU: zie hoofdstukken 9 en10.



Figuur 8.3: Neuraal netwerk van KIKS.

Deze 24 matrices worden dan samengesteld tot een tensor, de zogenaamde ***feature map*** (zie Figuur 8.4). Deze feature map dient dan als input voor de volgende laag.

In elke laag van het convolutionele neurale netwerk worden de tensoren door de convoluties getransformeerd in een nieuwe representatie van de gegevens.



Figuur 8.4: Feature map.

De feature map die de uitvoer was van het convolutionele netwerk, wordt gegeven aan een netwerk met dense layers (zie tweede deel van het netwerk in Figuur 8.3).

In hoofdstuk 10 zal blijken dat de invoer daar gebeurt met 1D-tensoren. Daarom voert men op de feature map eerst een operatie uit die alle elementen van de feature map achter elkaar plaatst: dat gebeurt met `flatten()` of met `reshape()`.

Zie ook hoofdstuk 11.

Voorbeeld:

```
>>> afb_sobel.shape
(5, 5)
>>> afb_sobel.flatten()
array([150, 160, 100, 25, 20, 80, 155, 85,
       5, 0, -10, -15, -15, -5, 0, -80,
       -160, -100, -25, -20, -140, -215,
       -230, -105, -105])
>>> afb_sobel.reshape(5*5)
array([150, 160, 100, 25, 20, 80, 155, 85,
       5, 0, -10, -15, -15, -5, 0, -80,
       -160, -100, -25, -20, -140, -215,
       -230, -105, -105])
```

Notebook (facultatief)

- Meer uitleg over tensoren vind je in de notebook `Tensoren.ipynb` in de map `IntroductiePython`.

8.6 Succes van deep learning

Deep learning heeft zoveel succes omdat het bij veel problemen, bv. beeldherkenning, betere resultaten boekt dan de klassieke machine learning-netwerken, maar ook omdat het een belangrijk aspect ervan, de *feature engineering*, volledig automatiseert (Chollet, 2018). Men moet niet meer zelf op zoek naar geschikte filters om relevante kenmerken uit de data te extraheren. **Het systeem gaat zelf op zoek naar bruikbare filters.**

Voor meer uitleg over deep learning: zie paragraaf 7.3

Samengevat

Om huidmondjes te herkennen, gebruikt KIKS een diep neuraal netwerk dat bestaat uit een convolutioneel netwerk, gevolgd door een netwerk met *dense layers*.

Samengevat - computerwetenschappen

In convolutionele netwerken werken filters in op afbeeldingen om er kenmerken in op te sporen. Deze convoluties zijn lineaire filters: er wordt een lineaire combinatie genomen van de pixelwaarden van een pixel en zijn buren.
In één laag van het convolutionele netwerk gebeuren er meerdere convolutiebewerkingen. Elke convolutie levert een matrix op, waarop vervolgens een niet-lineaire activatiefunctie inwerkt en opnieuw een matrix oplevert. De matrices worden samengesteld tot een tensor, de *feature map*.
De uitvoer van elke laag is dus een *feature map* die dient als input voor de volgende laag. In elke convolutionele laag wordt de ingevoerde tensor getransformeerd in een nieuwe representatie van de gegevens.

Na dit hoofdstuk weet je waarvoor een convolutie gebruikt wordt.
Bovendien kan je zelf de convolutiebewerking manueel uitvoeren.

HOOFDSTUK
NEGEN

RELU EN MAX POOLING

9.1 ReLU en max pooling in een convolutioneel neuraal netwerk

Om de huidmondjes op een microfoto te detecteren, doen de onderzoekers van KIKS een beroep op een diep neuraal netwerk dat bestaat uit een convolutioneel neuraal netwerk gevolgd door een netwerk met dense layers.

Het is een *Sequential model*, een model dat bestaat uit aaneengeschakelde lagen. Het convolutionele netwerk bezit enkele convolutionele lagen, afgewisseld met *max pooling*-lagen; het netwerk dat erop volgt, is er een met dense layers.

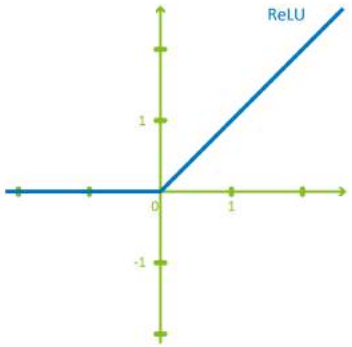
Zie ook hoofdstukken 8 en 11.

De convoluties passen filters toe op de inputtensoren. De convoluties worden gevolgd door een ReLU-functie met aansluitend een *max pooling*-operatie.

De ReLU-functie wordt als volgt gedefinieerd:

$$\text{ReLU} : \mathbb{R} \to \mathbb{R} : x \mapsto \begin{cases} 0, \text{ als } x < 0 \\ x, \text{ als } x \geq 0. \end{cases}$$

Kort gezegd: ReLU(x) = max(0,x).

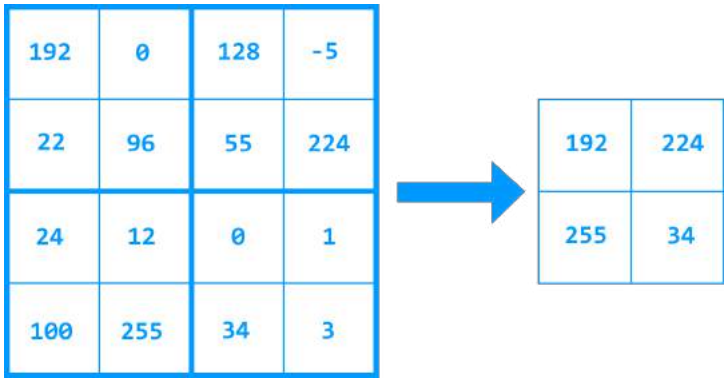


Figuur 9.1: Grafiek van de ReLU-functie.

Max pooling neemt uit elk vak van $2x2$ de grootste waarde. Dit wordt geïllustreerd in Figuur 9.2.

ReLU zorgt ervoor dat de kenmerken die het minst tot een goede classificatie leiden, worden verzwakt. ReLU zal er bv. bij grijswaardenafbeeldingen voor zorgen dat de lichte kleuren behouden blijven en de donkere kleuren worden afgevlakt (zie Figuur 9.3).

Max pooling zal de kenmerken die van belang zijn om tot een goede classificatie te komen, versterken en de andere kenmerken weglaten. Bovendien zorgt max pooling ervoor dat de tensoren kleiner worden, waardoor er minder rekenkracht nodig is door de computer. Door max pooling zal het netwerk ook minder snel overfitten. Men kan natuurlijk niet onbeperkt max



Figuur 9.2: Max pooling-operatie.

pooling-operaties toepassen, er moeten nog voldoende pixels overblijven.

Notebook

- Bekijk het effect van ReLU en max pooling in de notebook `ReLUMaxPooling.ipynb` in de map `IntroductieDeepLearning`.

Een voorbeeld:

- Het resultaat van ReLU op $A = \begin{bmatrix} 30 & -20 & 0 & -120 \\ -80 & 30 & 80 & 50 \\ 20 & -30 & -90 & -120 \end{bmatrix}$ is $\begin{bmatrix} 30 & 0 & 0 & 0 \\ 0 & 30 & 80 & 50 \\ 20 & 0 & 0 & 0 \end{bmatrix}$.
- Is $B = \begin{bmatrix} 30 & 20 & 130 & 120 & 50 & 50 \\ 80 & 130 & 80 & 50 & 120 & 40 \\ 20 & 30 & 90 & 120 & 100 & 100 \\ 50 & 90 & 30 & 30 & 10 & 10 \\ 50 & 40 & 120 & 120 & 50 & 130 \\ 130 & 120 & 130 & 10 & 100 & 40 \end{bmatrix}$, dan geeft max pooling $\begin{bmatrix} 130 & 130 & 120 \\ 90 & 120 & 100 \\ 130 & 130 & 130 \end{bmatrix}$.

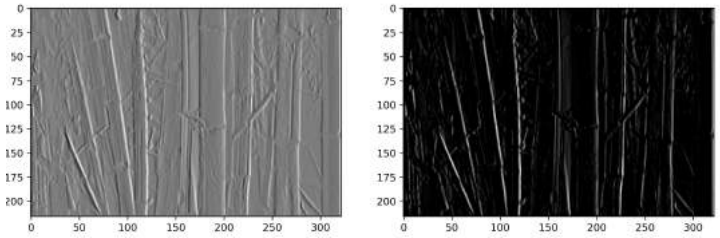
9.2 Effect van ReLU

Herneem het voorbeeld met de foto van de bamboeplant uit hoofdstuk 8. Je kan de verticale lijnen detecteren met een convolutie.

Je maakt daarbij gebruik van de filter $\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$.

Erna pas je ReLU toe op de gefilterde foto.

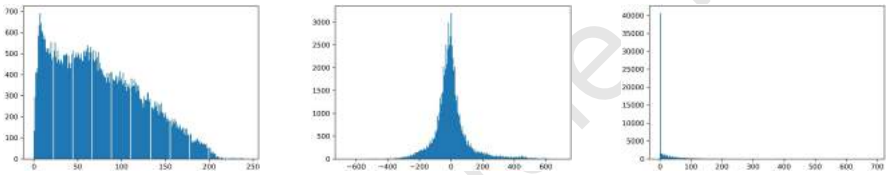
Het resultaat wordt getoond in Figuur 9.3.



Figuur 9.3: ReLU toepassen op de gefilterde foto.

De oorspronkelijke grijswaardenafbeelding `bamboe` heeft elementen tussen 0 en 255. Door het uitvoeren van de convolutie werden waarden bekomen die groter konden zijn en zelfs negatief. De ReLU-functie heeft alle negatieve waarden op nul gezet en de andere behouden.

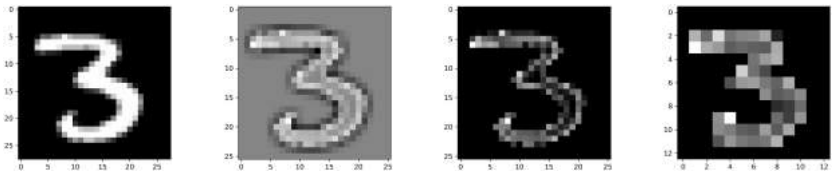
Na de convolutie wordt de waarde 689 geïnterpreteerd als wit en de waarde -665 als zwart. Alle grijswaarden ertussen worden verhoudingsgewijs toegekend. De ReLU-functie zal dus alle waarden donkerder dan een soort 'middengrijs' zwart maken. De verdeling van de grijswaarden wordt weergegeven in Figuur 9.4.



Figuur 9.4:
Links: Originele foto.
Midden: Gefilterde foto.
Rechts: Gefilterd, ReLU.

9.3 Effect van max pooling

Beschouw een '3' uit de MNIST dataset. Bekijk het effect van een convolutie, gevolgd door ReLU en max pooling in Figuur 9.5.



Figuur 9.5: Convolutie met randfilter op '3' uit MNIST dataset, gevolgd door ReLU en max pooling.

De max pooling maakt de afbeelding vier keer kleiner.

Samengevat - computerwetenschappen

Om huidmondjes te herkennen, gebruikt KIKS een diep neuraal netwerk dat bestaat uit een convolutioneel netwerk, gevolgd door een netwerk met *dense layers*. Het model bestaat uit aaneengeschakelde lagen.
In elke laag passen de convoluties filters toe op de inputtensoren. Ze worden gevolgd door een ReLU-functie met aansluitend een *max pooling*-operatie.

ReLU zorgt ervoor dat de kenmerken die het minst tot een goede classificatie leiden, worden verzwakt. ReLU zal er bv. bij grijswaardenafbeeldingen voor zorgen dat de lichte kleuren behouden blijven en de donkere kleuren worden afgevlakt.
Max pooling zal de kenmerken die van belang zijn om tot een goede classificatie te komen, versterken en de andere kenmerken weglaten. Bovendien zorgt max pooling ervoor dat de tensoren kleiner worden, waardoor er minder rekenkracht nodig is door de computer.

HOOFDSTUK
TIEN

BASISCONCEPTEN VAN MACHINAAL LEREN IN DE PRAKTIJK

10.1 Misconcepties

Via onze smartphone en het internet speelt AI een, soms verdoken, rol in ons leven. Overheid, justitie, veiligheidsdiensten, banken en verzekeringsfirma's, zoekrobots, socialemediaplatformen ... laten hun werking steeds meer afhangen van zogenaamde geautomatiseerde beslissingen, m.a.w. beslissingen geautomatiseerd met *machine learning*. AI heeft zo steeds meer impact op onze maatschappij, en de manier waarop neurale netwerken getraind worden kan, als men hier niet bedacht op is, vooroordelen en stereotypen aanwezig in onze maatschappij, versterken.

Over machine learning-systemen bestaan echter veel **misverstanden**:

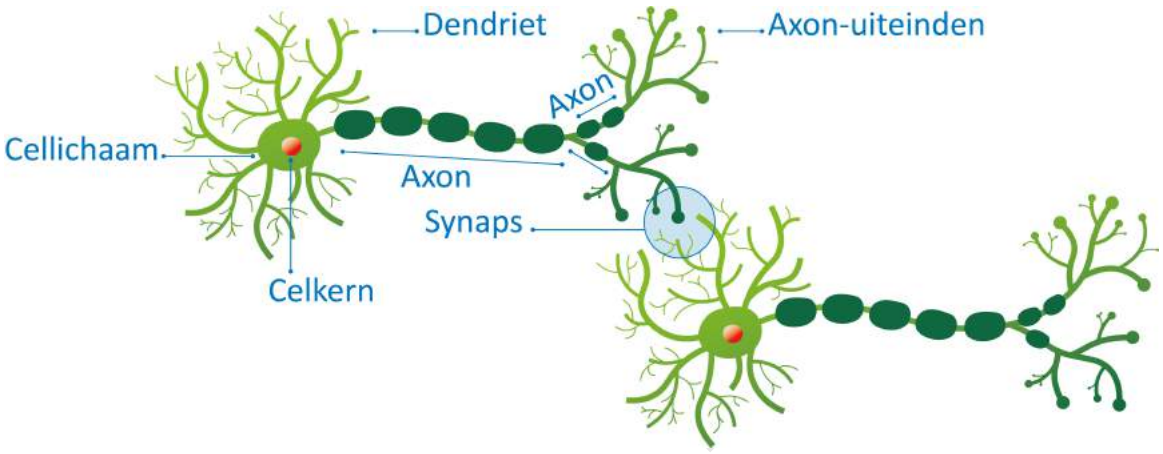
- De artificiële intelligentie die nu gebruikt wordt, staat zeer ver af van *general AI*. Het gaat om *machine learning*-systemen, gericht op een bepaalde taak, al dan niet geïmplementeerd in een robot; *narrow AI* dus.
- Omdat ze o.a. gebaseerd zijn op wiskundige principes, wordt verkeerdelijk gedacht dat deze systemen 100 % objectief beslissingen nemen.
- Een deep reinforcement learning-systeem dat in staat is de regerende schaakkampioen te verslaan, wordt al snel geacht veel slimmer te zijn dan die mens.
- De naam 'neurale netwerken' wekt de indruk dat deze systemen op dezelfde manier zouden werken als het menselijk brein.

10.2 Biologisch neuron

Een mens heeft ongeveer honderd miljard zenuwcellen of neuronen. De meeste bevinden zich in de hersenen of het ruggenmerg. Neuronen worden ook wel de bouwstenen van onze hersenen genoemd.

Een neuron bestaat uit dendrieten, een cellichaam en een axon. Het cellichaam bevat een celkern. Het axon mondt uit in de axoneinden en kan eventueel nog enkele vertakkingen hebben. Zoals elke cel is een neuron omgeven door een membraan. U ziet een afbeelding van twee neuronen in Figuur 10.1.

Bij een neuron in rust zijn er aan de buitenkant van de membraan meer positieve ionen dan aan de binnenkant, en aan de binnenkant zijn er meer negatieve ionen dan aan de buitenkant. Dit geeft een potentiaalverschil over de celmembraan: de rustpotentiaal.



Figuur 10.1: Biologische neuronen.

Een neuron ontvangt prikkels: via de dendrieten krijgt het signalen van andere neuronen.

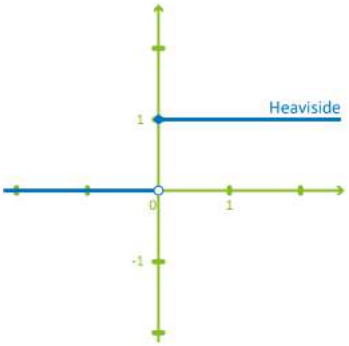
Als een neuron geprikkeld wordt, kunnen er zich, afhankelijk van de **sterkte van de prikkel**, meer of minder ionen verplaatsen door de membraan. Als het totaal aan signalen sterk genoeg is, verplaatsen er zich zoveel ionen door de membraan dat het potentiaalverschil voldoende gestegen is om een elektrische impuls te doen ontstaan. In dat geval is het neuron **geëxciteerd**. Sommige signalen werken de geëxciteerde toestand meer in de hand, andere temperen de excitatie. Men zou kunnen zeggen dat sommige verbindingen tussen neuronen sterker zijn dan andere.

Dus eens het potentiaalverschil een bepaalde **drempelwaarde** overschrijdt, veroorzaakt deze zogenaamde actiepotentiaal een elektrische impulsgeleiding: een elektrische impuls van het cellichaam door het axon naar de axoneinden. Aan het axoneinde komen neurotransmitters vrij, die op membraanreceptoren van een volgend neuron binden, waardoor in dat neuron nu ook een elektrische impulsgeleiding kan ontstaan. M.a.w. **het neuron stuurt op zijn beurt via het axon een signaal naar andere neuronen**.

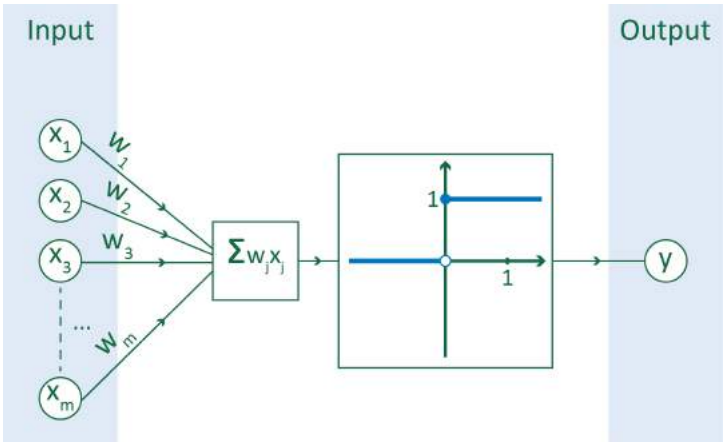
Na de impuls keert het neuron terug in rust.

10.3 Artificieel neuron

In 1943 ontwierpen McCulloch en Pitts een artificieel neuron, zoals voorgesteld in Figuur 10.3 (Haykin, 2009). De functie die er gebruikt wordt, is een **drempelwaardefunctie**, de Heaviside-functie H.



Figuur 10.2: Heaviside-functie.



Figuur 10.3: Artificieel neuron.

De Heaviside-functie wordt als volgt gedefinieerd:

$$H : \mathbb{R} \to \mathbb{R} : x \mapsto \begin{cases} 0, \text{ als } x < 0 \\ 1, \text{ als } x \geq 0. \end{cases}$$

De **inputwaarden die het neuron ontvangt van andere neuronen**, worden gewogen opgeteld; het **gewicht** bepaalt a.h.w. de sterkte van de verbinding tussen de neuronen. De som wordt onderworpen aan de niet-lineaire Heaviside-functie H. Als de som de drempelwaarde 0 bereikt of overschrijdt, krijgt het outputneuron y waarde 1, anders krijgt het waarde 0. De Heaviside-functie noemt men een activatiefunctie en ze bepaalt de toestand y van het neuron. Net als het biologisch neuron, **geëxciteerd** of niet geëxciteerd, kent het artificiële neuron twee toestanden.

Opdracht - biologisch en artifcieel neuron

Leg uit hoe het artificiële neuron geïnspireerd is op een biologisch neuron.

Frank Rosenblatt ging hierop verder in 1958. Hij bouwde een systeem opdat een artificieel neuron zou kunnen leren: het Perceptron, het eerste neurale netwerk.

Het biologische neuron diende voor wetenschappers zoals McCulloch, Pitts en Rosenblatt als inspiratie voor de ontwikkeling van de theorie rond neurale netwerken, maar het contrast met de werking van het artificiële neuron is groot: **in de hersenen gebeuren de processen continu en parallel; bij het artificiële neuron gebeuren die discreet en serieel**.

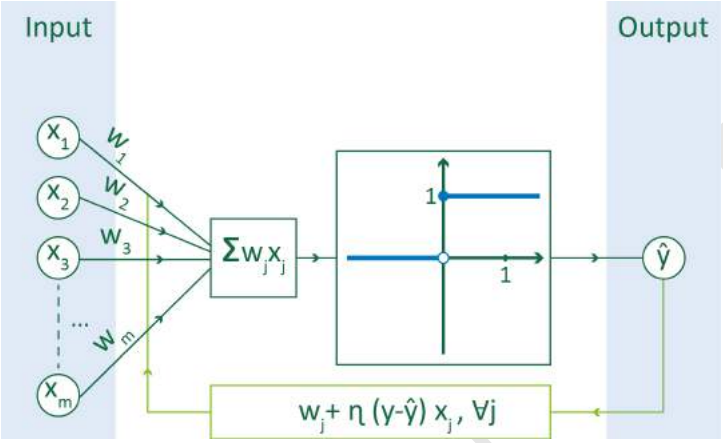
De neurale netwerken die hierna aan bod komen, hebben niet de bedoeling de structuur van het menselijk brein te imiteren.

10.4 Perceptron

Het Perceptron is een neuraal netwerk met twee lagen: een inputlaag en een outputlaag. De neuronen van de inputlaag zijn verbonden met het neuron in de outputlaag.

Het Perceptron beschikt over een algoritme om te leren. Het wordt getraind met gelabelde voorbeelden: een aantal inputpunten X_i met telkens een corresponderende output y_i, het label. Tussen de neuronen van de input- en outputlaag zijn er verbindingen met een bepaald gewicht. Gebaseerd op de gelabelde voorbeelden worden die gewichten gaandeweg aangepast.

Het Perceptron wordt schematisch voorgesteld in Figuur 10.4. $X_i(x_1, x_2, ..., x_m)$. **Het systeem kiest eerst willekeurige waarden**



Figuur 10.4: Het Perceptron. Activatiefunctie is Heaviside-functie met drempelwaarde 0.

voor de gewichten w_j.

Voor elk voorbeeld wordt nagegaan of de klasse $\hat{y}_i$ die het systeem voorspelt, wel overeenkomt met het gegeven label y_i. Indien $\hat{y}_i$ en y_i niet overeenkomen, dan worden de waarden van de gewichten aangepast. Men zegt dat het systeem leert.

Men vermeerdert daartoe elke w_j met een waarde Δw_j waarbij Δw_j afhangt van het verschil tussen y_i en $\hat{y}_i$, van x_j en van de leersnelheid (*learning rate*).

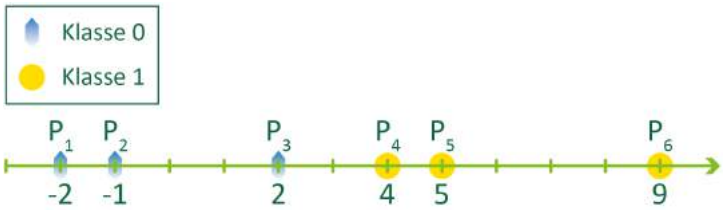
Het systeem zal de gegeven voorbeelden meerdere keren doorlopen tot voor alle voorbeelden het juiste label wordt voorspeld. Per keer dat alle voorbeelden doorlopen worden, spreekt men van een *epoch*.

De learning rate of de leersnelheid is een hyperparameter van het netwerk. Het is een getal, gekozen tussen 0 en 1, dat eigenlijk bepaalt hoe bruusk of hoe voorzichtig de gewichten aangepast worden; de leersnelheid zal dus mee bepalen hoe snel het systeem leert. De leersnelheid wordt genoteerd met de Griekse letter η (èta).

Onthoud: de gelabelde voorbeelden en de learning rate worden gekozen door de mens, niet door het neurale netwerk.

10.5 Perceptron met aangepaste drempelwaarde

Beschouw het voorbeeld uit Figuur 10.5.



Figuur 10.5: Punten die behoren tot twee klassen. De abscis x van elk punt is af te lezen op de figuur.

Voor dit voorbeeld is het interessanter om 3 als de drempelwaarde van de activatiefunctie te nemen, i.p.v. 0.

Immers: de punten rechts van 3 behoren tot een andere klasse dan de punten links van 3. Je zou dus een scheiding kunnen plaatsen op 3.

Om deze nieuwe drempelwaarde 3 i.p.v. 0 te realiseren, voeg je een extra inputneuron, 1, en een extra gewicht, w_0, toe. Hier geldt dan dat $w_0 = -3$.

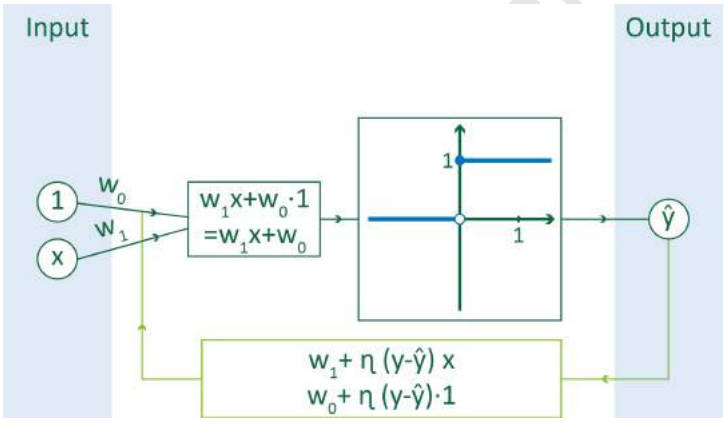
$x - 3$ is een gewogen som van x en 1 met gewichten resp. 1 en -3. In de wiskunde noemt men dat een lineaire combinatie van x en 1:

$$x - 3 = 1 \cdot x + (-3) \cdot 1.$$

$(-3) \cdot 1 + 1 \cdot x = w_0 x_0 + w_1 x_1$ met $w_0 = -3$ en $w_1 = 1$. Men noemt die 3 de (wiskundige) *bias* van de invoerlaag van het netwerk.

Deze werkwijze wordt voorgesteld in Figuur 10.6. Men voegt een (input)neuron toe met waarde 1 (dit toegevoegd neuron wordt genoteerd als x_0) en een extra gewicht (genoteerd als w_0).

Onthoud dat een scheiding met vergelijking $x = 3$ overeenkomt met een gewicht $w_0 = -3$.



Figuur 10.6: Neuraal netwerk met 2 inputneuronen. Het extra neuron x_0 speelt een cruciale rol.

Men werkt dan met input $X_i(1, x)$ met $x_0 = 1$, $x_1 = x \in \mathbb{R}$, label $y_i \in \{0,1\}$, output $\hat{y}_i \in \{0,1\}$ voor $i = 1, ..., n$ en $\overrightarrow{W}(w_0, w_1)$, met $w_j \in \mathbb{R}, j = 0, 1$.

Voor het punt P_1 uit Figuur 10.5 is de input $X_1(1, -2)$ en het label $y_1 = 0$; $x_0 = 1$ en $x_1 = -2$.

De berekening die wordt uitgevoerd door het netwerk, kan worden voorgesteld in matrixgedaante.

Notebook (facultatief)

- Hoe je rekent met matrices, wordt getoond in de notebook `Tensoren.ipynb` in de map `IntroductiePython`.

Stel dat $X = \begin{bmatrix} 1 \\ x \end{bmatrix}$ en $W = \begin{bmatrix} -3 \\ 1 \end{bmatrix}$. Dan is

$$W^T \cdot X = \begin{bmatrix} -3 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ x \end{bmatrix} = \begin{bmatrix} -3 \cdot 1 + 1 \cdot x \end{bmatrix} = \begin{bmatrix} x - 3 \end{bmatrix}.$$

Dit wordt gevolgd door $H(x-3)$ met als resultaat $\hat{y}$.

Of meer algemeen:

$$X = \begin{bmatrix} 1 \\ x \end{bmatrix}, W = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix},$$

$$W^T \cdot X = \begin{bmatrix} w_0 & w_1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ x \end{bmatrix}$$

$$= \begin{bmatrix} w_0 \cdot 1 + w_1 \cdot x \end{bmatrix},$$

gevolgd door $H(w_0 + w_1 x) = \hat{y}$.

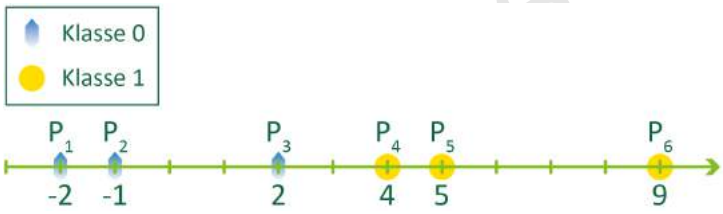
De waarde van $\hat{y}$ hangt af van het feit of $w_0 + w_1 x > 0$ of $w_0 + w_1 x < 0$ is. Dus de vergelijking $w_0 + w_1 x = 0$ bepaalt de scheiding.

10.6 1D-voorbeeld van de training van het Perceptron

Opdracht - training Perceptron

Je zal nu zelf de training van het Perceptron manueel uitvoeren.

Beschouw het voorbeeld uit Figuur 10.7.



Figuur 10.7: Punten die behoren tot twee klassen.

Er zijn zes voorbeelden waar het Perceptron uit kan leren: -2 uit klasse 0, -1 uit klasse 0, 2 uit klasse 0, 4 uit klasse 1, 5 uit klasse 1 en 9 uit klasse 1. Het is de mens die deze gelabelde voorbeelden kiest.

Bijvoorbeeld voor het punt P_1 geldt: $x = -2$ en de klasse $y = 0$, m.a.w. $x_0 = 1, x_1 = -2$ en $y = 0$.

Noteer alvast de waarden van deze variabelen voor de andere punten.

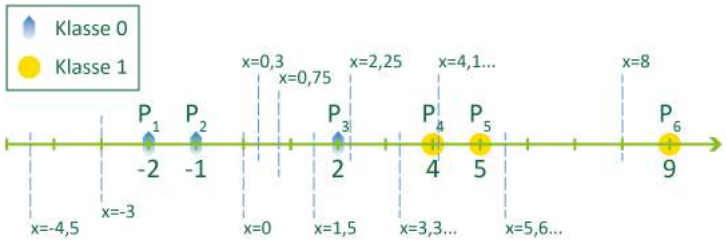
Er worden nu twee mogelijke scenario's besproken, elk met een gekozen leersnelheid, en waarbij vertrokken wordt van bepaalde waarden voor de gewichten w_0 en w_1.

10.6.1 $\eta = 0{,}1$ en $w_0 = -8, w_1 = 1$

De learning rate wordt vastgelegd op $\eta = 0{,}1$.

Het systeem initialiseert de waarden van de gewichten op een willekeurige manier, bv. $w_0 = -8$ en $w_1 = 1$. De voorlopige scheidingslijn ligt dan op 8 en heeft dus als vergelijking: $x = 8$. Hierdoor

zullen slechts 4 punten aan de juiste kant van de scheidingslijn liggen, nl. P_1, P_2, P_3 en P_6 (zie Figuur 10.8).



Figuur 10.8: Vier punten correct geclassificeerd.

Je kan de tabel gebruiken om het leren van het Perceptron neer te schrijven met initiële waarden $w_0 = -8$ en $w_1 = 1$ en $\eta = 0{,}1$. Verder vind je wat uitleg bij de tabel.

Punt	Klasse y	w_0	w_1	x_0	x_1	$w_0x_0 + w_1x_1$	>,< 0	Voorspelde klasse $\hat{y}$	$y = \hat{y}$	Gewichten aanpassen?
P_1	0	-8	1	1	-2	-10	<0	0	ja	nee
P_2	0	-8	1	1						nee
P_3	0	-8	1	1						
P_4	1	-8	1	1		-4	<0	0	nee	ja
P_5	1	-7,9	1,4							ja
P_6		-7,8	1,9							nee

Tabel 10.1: Eerste epoch.

Het systeem begint aan een eerste epoch. Het overloopt elk voorbeeldpunt.

P_1 met $y = 0, x_0 = 1, x_1 = -2$:

$$-2 \xrightarrow{w_1x_1+w_0} 1.(-2) + (-8) = -10 \xrightarrow[-10<0]{H} 0$$

dus $\hat{y} = 0$ en $\hat{y} = y$.

De gewichten moeten niet worden aangepast.

Reken na dat ook voor P_2 en P_3 de gewichten niet moeten worden aangepast.

P_4 met $y = 1, x_0 = 1, x_1 = 4$:

$$4 \xrightarrow{w_1x_1+w_0} 1.4 + (-8) = -4 \xrightarrow[-4<0]{H} 0$$

dus $\hat{y} = 0$ en $\hat{y} \neq y$.

De gewichten moeten wel worden aangepast.

Het gewicht w_j wordt vervangen door

$$w_j + \eta \cdot (y - \hat{y}) \cdot x_j,$$

dus

$$w_0 = -8 + 0{,}1 \cdot (1 - 0) \cdot 1 \Leftrightarrow w_0 = -8 + 0{,}1 \Leftrightarrow w_0 = -7{,}9$$

Hoe groter de fout, met name het verschil tussen het label y en de voorspelde waarde $\hat{y}$, hoe groter de aanpassing die men zal doen.

en

$$w_1 = 1 + 0{,}1 \cdot (1 - 0) \cdot 4 \Leftrightarrow w_1 = 1 + 0{,}4 \Leftrightarrow w_1 = 1{,}4.$$

De vergelijking $w_0 + w_1 x = 0$ bepaalt de scheiding.

Merk op dat de scheidingsrechte naar de juiste kant is opgeschoven: naar links! De voorlopige scheidingslijn heeft nu als vergelijking:

$$1{,}4x - 7{,}9 = 0 \Leftrightarrow x = 5{,}642....$$

P_5 met $y = 1, x_0 = 1, x_1 = 5$:

$$5 \xrightarrow{w_1 x_1 + w_0} 1{,}4 \cdot 5 + (-7{,}9) = -0{,}9 \xrightarrow[-0{,}9<0]{H} 0$$

dus $\hat{y} = 0$ en $\hat{y} \neq y$.

De gewichten moeten opnieuw worden aangepast.

$$w_0 = -7{,}9 + 0{,}1 \cdot (1 - 0) \cdot 1 \Leftrightarrow w_0 = -7{,}9 + 0{,}1 \Leftrightarrow w_0 = -7{,}8$$

en

$$w_1 = 1{,}4 + 0{,}1 \cdot (1 - 0) \cdot 5 \Leftrightarrow w_1 = 1{,}4 + 0{,}5 \Leftrightarrow w_1 = 1{,}9.$$

Merk op dat de scheidingsrechte naar de juiste kant is opgeschoven: naar links! De voorlopige scheidingslijn heeft nu als vergelijking:

$$1{,}9x - 7{,}8 = 0 \Leftrightarrow x = 4{,}157....$$

Reken na dat P_6 bij de juiste klasse wordt ingedeeld, dus voor P_6 moeten de gewichten niet worden aangepast.

Alle voorbeelden zijn nu één keer doorlopen. Men zegt dat de eerste epoch voltooid is.

Nu onderneemt het systeem een tweede epoch. Het doorloopt opnieuw alle voorbeelden, te beginnen met P_1.

Punt	Klasse y	w_0	w_1	x_0	x_1	$w_0x_0 + w_1x_1$	$>,<0$	Voorspelde klasse $\hat{y}$	$y = \hat{y}$	Gewichten aanpassen?
P_1	0	-7,8	1,9	1	-2					
P_2										
P_3										
P_4										
P_5		-7,7	2,3							
P_6										

Tabel 10.2: Tweede epoch.

Uitleg:

$$-2 \xrightarrow{w_1 x_1 + w_0} 1{,}9 \cdot (-2) + (-7{,}8) = -11{,}6 \xrightarrow[-11{,}6<0]{H} 0$$

dus $\hat{y} = 0$ en $\hat{y} = y$.

P_1 wordt bij de juiste klasse ingedeeld.

Reken na dat dat ook geldt voor P_2 en P_3.

Opdracht - training Perceptron

- Vul de tabel aan.
- Doe nu zelf verder met de andere punten.
- Onderneem zoveel epochs als nodig: de laatste epoch is de epoch waarbij alle punten bij de juiste klasse worden ingedeeld en waarbij er dus geen aanpassingen aan de gewichten meer moeten gebeuren.

In de derde epoch zijn de punten juist geclassificeerd.

Punt	Klasse y	w_0	w_1	x_0	x_1	$w_0x_0 + w_1x_1$	$>,<0$	Voorspelde klasse $\hat{y}$	$y = \hat{y}$	Gewichten aanpassen?
P_1										
P_2										
P_3										
P_4										
P_5										
P_6										

Tabel 10.3: Derde epoch.

10.6.2 $\eta = 0{,}3$ en $w_0 = 0, w_1 = 1$

Opdracht - training Perceptron met grotere learning rate

- Doe nog eens hetzelfde maar met een grotere leersnelheid, nl. learning rate $\eta = 0{,}3$ en andere beginwaarden voor de gewichten. De gewichten w_0 en w_1 worden nu geïnitialiseerd op resp. 0 en 1. De voorlopige scheidingslijn heeft dan als vergelijking: $x = 0$.
- Je zal merken dat de scheidingslijn wel eens naar de verkeerde kant opschuift, maar je moet dan gewoon doorgaan en uiteindelijk komt het goed.

10.7 Het Perceptron-algoritme

```
BEGIN
    HERHAAL tot alle punten bij de juiste klasse worden ingedeeld
        VOOR ELK PUNT met corresponderend label in de dataset
            VERGELIJK het label met de voorspelling
            ALS de voorspelling fout is DAN
                pas de gewichten W⃗ aan naar W⃗ + η(y − ŷ) · X⃗
            ANDERS (dus als de voorspelling juist is) DOE
                niets
EINDE
```

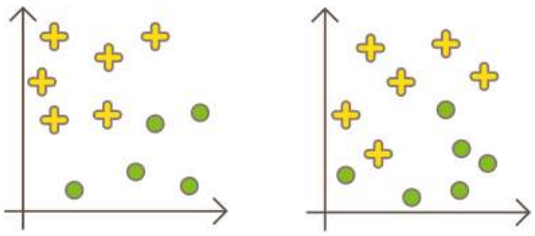
Punt	Klasse y	w_0	w_1	x_0	x_1	$w_0x_0 + w_1x_1$	$>,<0$	Voorspelde klasse $\hat{y}$	$y = \hat{y}$	Gewichten aanpassen?
P_1										
P_2										
P_3										
P_4										
P_5										
P_6										

Punt	Klasse y	w_0	w_1	x_0	x_1	$w_0x_0 + w_1x_1$	$>,<0$	Voorspelde klasse $\hat{y}$	$y = \hat{y}$	Gewichten aanpassen?
P_1										
P_2										
P_3										
P_4										
P_5										
P_6										

Het Perceptron-algoritme kan toegepast worden om lineair scheidbare data te classificeren. Het is dus maar beperkt bruikbaar. In de praktijk zal het meestal niet mogelijk zijn alle data in de juiste klasse in te delen. Als men voor zulke data die niet lineair scheidbaar zijn, toch kiest voor dit algoritme om de data te scheiden, dan zal het algoritme blijven doorgaan. Men heeft dan te maken met een oneindige lus.

Er zullen immers steeds punten zijn waarvoor een foute voorspelling wordt gemaakt.

Figuur 10.9 toont het verschil tussen data die lineair scheidbaar zijn en data die dat niet zijn.



Figuur 10.9: Wel of niet lineair scheidbaar?

10.8 2D-voorbeeld van het Perceptron: classificatie met de Iris dataset

De Iris dataset werd in 1936 door de Brit Ronald Fischer gepubliceerd in 'The use of multiple measurements in taxonomic problems' (Fischer, 1936; Dua & Taniskidou, 2017). De dataset betreft drie soorten irissen: *Iris setosa*, *Iris virginica* en *Iris versicolor* (zie Figuren 10.10, 10.11 en 10.12).

Punt	Klasse y	w_0	w_1	x_0	x_1	$w_0x_0 + w_1x_1$	$>,< 0$	Voorspelde klasse $\hat{y}$	$y = \hat{y}$	Gewichten aanpassen?
P_1										
P_2										
P_3										
P_4										
P_5										
P_6										

Punt	Klasse y	w_0	w_1	x_0	x_1	$w_0x_0 + w_1x_1$	$>,< 0$	Voorspelde klasse $\hat{y}$	$y = \hat{y}$	Gewichten aanpassen?
P_1										
P_2										
P_3										
P_4										
P_5										
P_6										

Punt	Klasse y	w_0	w_1	x_0	x_1	$w_0x_0 + w_1x_1$	$>,< 0$	Voorspelde klasse $\hat{y}$	$y = \hat{y}$	Gewichten aanpassen?
P_1										
P_2										
P_3										
P_4										
P_5										
P_6										

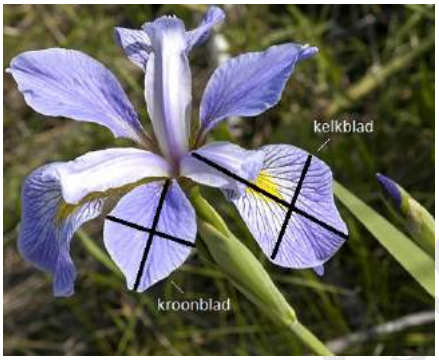
Punt	Klasse y	w_0	w_1	x_0	x_1	$w_0x_0 + w_1x_1$	$>,< 0$	Voorspelde klasse $\hat{y}$	$y = \hat{y}$	Gewichten aanpassen?
P_1										
P_2										
P_3										
P_4										
P_5										
P_6										

Figuur 10.10:
Iris setosa
(Binek, 2019).

Figuur 10.11:
Iris virginica
(Mayfield, 2007).

Figuur 10.12:
Iris versicolor
(Langlois, 2019).

Fischer kon de soorten van elkaar onderscheiden afgaande op vier kenmerken: de lengte en de breedte van de kelkbladen en de kroonbladen (zie Figuur 10.13).



Figuur 10.13: Kroon- en kelkbladen.

Dit leidt tot een onderzoeksvraag: lukt het om met het Perceptron twee soorten irissen van elkaar te onderscheiden op basis van twee kenmerken: de lengte van het kroonblad en de lengte van het kelkblad?

Notebook - classificatie met het Perceptron

- In de notebook `Iris.ipynb` in de map `IntroductieMachineLearning` wordt het Perceptron toegepast om te classificeren: het scheiden van twee soorten irissen op basis van twee kenmerken. De twee klassen kunnen lineair van elkaar worden gescheiden.
- De onderzoeksvraag die gesteld wordt in deze notebook, is: kunnen de *Iris setosa* en de *Iris versicolor* van elkaar onderscheiden worden op basis van twee kenmerken?
- Er wordt getoond welk effect de aanpassingen op de gewichten hebben op de ligging van de scheidingsrechte.
- Herken je het Perceptron-algoritme in de gegeven code?

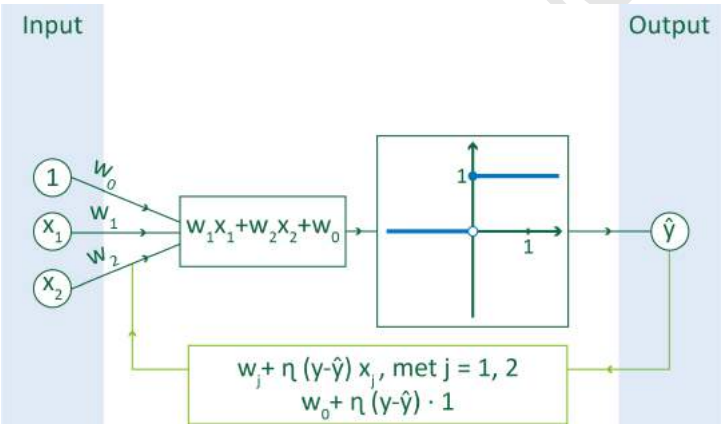
Notebook - standaardiseren

- Vooraleer men machine learning-technieken gaat toepassen op de data, zal men de data eerst standaardiseren of normaliseren. Dat levert een grafische voorstelling op van de data die een goed beeld geeft van de correlatie tussen de variabelen en kleinere getallen om mee te rekenen. Meer uitleg over standaardiseren en normaliseren vind je in de notebook `Standaardiseren.ipynb` in de map `IntroductieMachineLearning`.

In de notebook `Iris.ipynb` gebruikt men de kenmerken lengte kelkblad en lengte kroonblad. De twee kenmerken worden weergegeven door x_1 en x_2, de soort iris door het label y: label $y = 0$ (voor de *setosa*) en label $y = 1$ (voor de *versicolor*).

Elk datapunt is dus een punt (x_1,x_2) met een corresponderend label y.

Omdat men werkt met twee kenmerken, zijn er al twee inputneuronen. Opnieuw voegt men een extra neuron toe. Er moeten nu dus drie gewichten worden geïnitialiseerd, ze worden aangepast op basis van de acht gelabelde voorbeelden (zie Figuur 10.14).



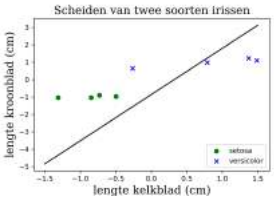
Figuur 10.14: Perceptron-neuraal netwerk met drie neuronen.

De werkwijze, nu in twee dimensies, is dezelfde als daarnet in één dimensie (paragraaf 10.6). Het verschil is dat er nu drie gewichten zijn om aan te passen i.p.v. twee.

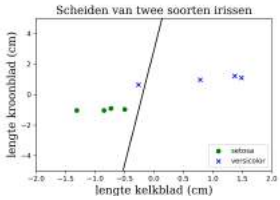
Na initialisatie van de gewichten is de voorlopige scheidingsrechte bv. de rechte uit Figuur 10.15.

In de loop van de training 'verplaatst' de rechte zich. Na enige epochs bekomt men de rechte uit Figuur 10.16. Na de training bekomt men een scheiding zoals in Figuur 10.17.

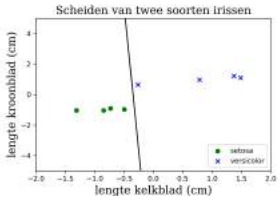
De grafieken in de Figuren 10.15, 10.16 en 10.17 bevatten gestandaardiseerde waarden.



Figuur 10.15: Initialisatie van de scheidingslijn.



Figuur 10.16: Rechte in de loop van de training.



Figuur 10.17: Scheidingsrechte na training.

Onthoud - binaire classificatie

- Onthoud zeker dat de initiële scheidingsrechte willekeurig werd gekozen. De coëfficiënten in de vergelijking van de rechte spelen de rol van de gewichten die worden aangepast door de gelabelde voorbeelden aan de scheidingsrechte te toetsen.

Waarom werkt het Perceptron-algoritme? (Facultatief)

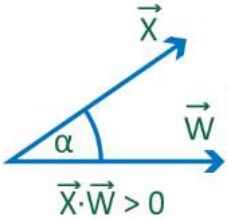
Scalair product

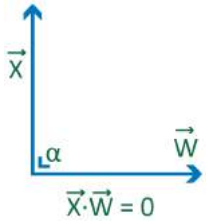
$w_1x_1 + w_2x_2 + w_0$ kan men interpreteren als het 'scalair product' van twee driedimensionale vectoren $\overrightarrow{W}(w_0,w_1,w_2)$ en $\overrightarrow{X}(1,x_1,x_2)$:

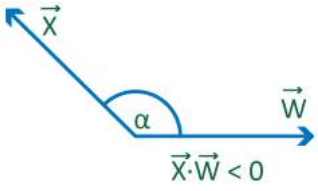
$$\overrightarrow{W} \cdot \overrightarrow{X} = w_0 \cdot 1 + w_1 \cdot x_1 + w_2 \cdot x_2 \quad \text{in analytische gedaante,}$$
$$\overrightarrow{W} \cdot \overrightarrow{X} = \|\overrightarrow{W}\| \cdot \|\overrightarrow{X}\| \cos(\overrightarrow{W}, \overrightarrow{X}) \quad \text{in meetkundige gedaante.}$$

Opschuiven naar de juiste klasse

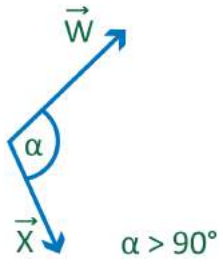
Als $\overrightarrow{W}$ en $\overrightarrow{X}$ loodrecht staan op elkaar, is $\overrightarrow{W} \cdot \overrightarrow{X} = 0$.
Als de hoek tussen de vectoren scherp is, is $\overrightarrow{W} \cdot \overrightarrow{X} > 0$ en als de hoek tussen de vectoren stomp is, is $\overrightarrow{W} \cdot \overrightarrow{X} < 0$.
Als men deze vectoren in hetzelfde punt laat aangrijpen, dan liggen hun representanten in eenzelfde vlak.





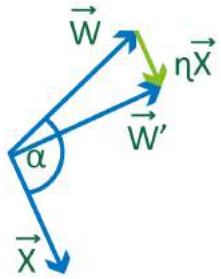


Als een punt wordt ingedeeld bij de foute klasse, dan worden de gewichten, dus $\overrightarrow{W}$, aangepast.
Stel dat (x_1, x_2) behoort tot de klasse met label 1. Dan is het vereist dat $\overrightarrow{W} \cdot \overrightarrow{X} > 0$ om het punt (x_1, x_2) in te delen bij de juiste klasse. Indien de hoek tussen de vectoren echter stomp is en de situatie uit Figuur 10.18 zich voordoet, dan is $\overrightarrow{W} \cdot \overrightarrow{X} < 0$ en wordt het punt (x_1, x_2) ingedeeld bij de verkeerde klasse. M.a.w. $y = 1, \hat{y} = 0, y - \hat{y} = 1$ en $\eta(y - \hat{y}) > 0$.



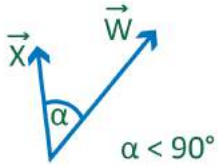
Figuur 10.18: Stompe hoek tussen beide vectoren.

Als $\overrightarrow{W}$ aangepast wordt naar $\overrightarrow{W} + \eta(y - \hat{y}) \cdot \overrightarrow{X}$, dan wordt de hoek tussen $\overrightarrow{W}$ en $\overrightarrow{X}$ kleiner (zie Figuur 10.19). Er wordt a.h.w. opgeschoven naar de juiste klasse.



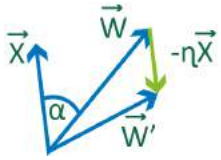
Figuur 10.19: Opschuiven naar klasse 1.

Stel dat (x_1, x_2) behoort tot de klasse met label 0. Dan zou $\overrightarrow{W} \cdot \overrightarrow{X} < 0$ moeten zijn om het punt (x_1, x_2) in te delen bij de juiste klasse. Indien de hoek tussen de vectoren echter scherp is en de situatie uit Figuur 10.20 zich voordoet, dan is $\overrightarrow{W} \cdot \overrightarrow{X} > 0$ en wordt het punt (x_1, x_2) ingedeeld bij de verkeerde klasse. M.a.w. $y = 0, \hat{y} = 1, y - \hat{y} = -1$ en $\eta(y - \hat{y}) < 0$.



Figuur 10.20: Scherpe hoek tussen beide vectoren.

Als $\overrightarrow{W}$ aangepast wordt naar $\overrightarrow{W} + \eta(y - \hat{y}) \cdot \overrightarrow{X}$, dan wordt de hoek tussen $\overrightarrow{W}$ en $\overrightarrow{X}$ groter (zie Figuur 10.21). Er wordt a.h.w. opgeschoven naar de juiste klasse.



Figuur 10.21: Opschuiven naar klasse 0.

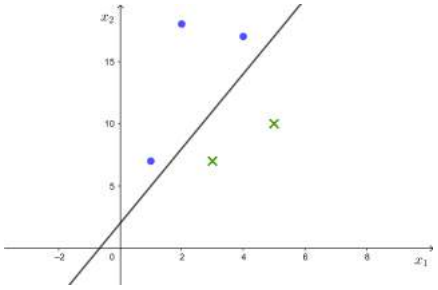
Wat is het verband tussen $\overrightarrow{W} \cdot \overrightarrow{X} < 0$ en de gezochte scheidingslijn?

Stel dat het ML-systeem getraind is en dat de gewichten uiteindelijk bepaald zijn door het ML-systeem: de waarde van $w_0 \cdot 1 + w_1 \cdot x_1 + w_2 \cdot x_2$ bepaalt dan bij welke klasse een punt (x_1, x_2) wordt ingedeeld. De waarde kan '> 0, < 0 of = 0' zijn en de 'scheiding' van beide klassen gebeurt op '= 0'.

$$w_0 \cdot 1 + w_1 \cdot x_1 + w_2 \cdot x_2 = 0 \Leftrightarrow w_2 \cdot x_2 = -w_1 \cdot x_1 - w_0$$

In het geval dat $w_2 \neq 0$, is $x_2 = -\frac{w_1}{w_2} \cdot x_1 - \frac{w_0}{w_2}$ of m.a.w. de scheidingslijn heeft als vergelijking

$y = -\frac{w_1}{w_2} \cdot x - \frac{w_0}{w_2}$. In het geval dat $w_2 = 0$, is de scheidingslijn verticaal met vergelijking $x = -\frac{w_0}{w_1}$.



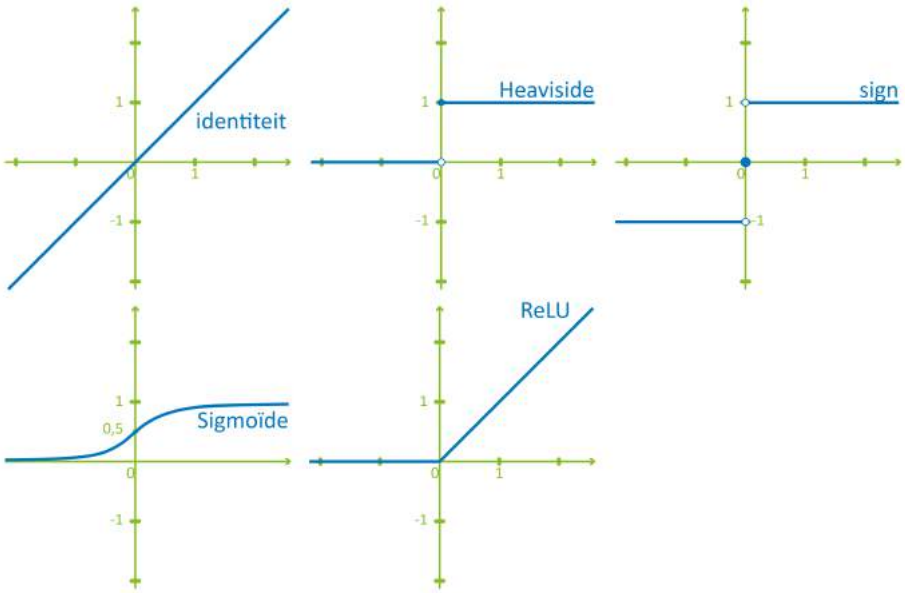
Figuur 10.22: Scheidingslijn.

(Rijksuniversiteit Groningen, 2005)

10.9 Activatiefuncties

Activatiefuncties

De Heaviside-functie (drempelwaardefunctie), de identiteitsfunctie, de sign-functie, de sigmoïde en de ReLU-functie zijn niet-lineaire functies. Ze worden vaak aangewend als activatiefunctie in een neuraal netwerk.



Figuur 10.23: Activatiefuncties.

10.10 Binaire classificatie

10.10.1 Sigmoïde als activatiefunctie

Soms zijn twee klassen **niet volledig lineair scheidbaar**, maar slechts op enkele punten na. Het Perceptron is voor dergelijke klassen niet zo geschikt.

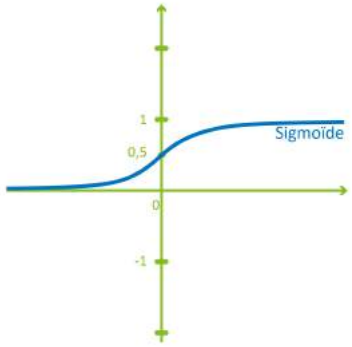
Voor het Perceptron-algoritme, zie paragraaf 10.7. De activatiefunctie van het Perceptron is de Heaviside-functie.

In dat geval zal men binair classificeren met *logistic regression* (Nerbonne, 2004)**.**

Als activatiefunctie wordt dan gebruikgemaakt van de **sigmoïde-functie**, een reële functie met voorschrift:

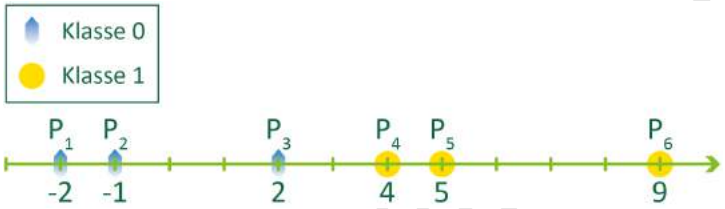
$$\sigma : \mathbb{R} \to \mathbb{R} \text{ met } \sigma(x) = \frac{1}{1+e^{-x}}.$$

Men spreekt ook van een logistieke functie. De output is een waarde $\hat{y}$ die weergeeft hoe zeker het model is dat de input tot de klasse met label 1 behoort. De zekerheid dat de input tot de andere klasse behoort, is dan $1 - \hat{y}$.

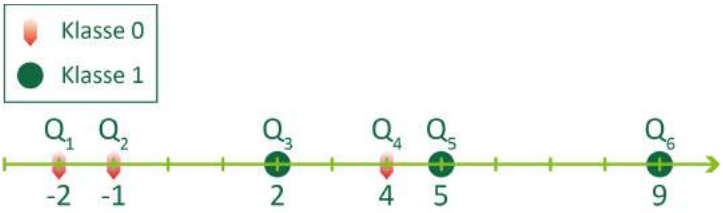


Figuur 10.24: Sigmoïde-functie.

Bekijk het voorbeeld in 1D met slechts één gewicht w (Figuur 10.25, paragraaf 10.6) en een tweede geval ook met één gewicht w (Figuur 10.26).



Figuur 10.25: Eerste geval: punten die behoren tot twee lineair scheidbare klassen.



Figuur 10.26: Tweede geval: punten die behoren tot twee klassen die niet lineair scheidbaar zijn.

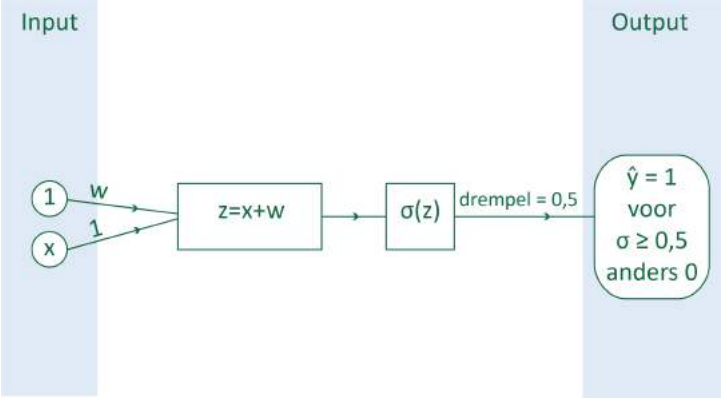
Voorbeelden zijn geïnspireerd op een blogpost van Daniel Godoy (Godoy, 2018).

Om de punten van elkaar te scheiden, dus onder te brengen in twee klassen, kan men een neuraal netwerk gebruiken zoals in Figuur 10.27.

Stel voor het tweede geval (Figuur 10.26) dat $w = 0$, dan gebeurt het volgende in het netwerk: $x \mapsto x$ als lineaire operatie, gevolgd door $x \mapsto \sigma(x)$ voor de activatie.

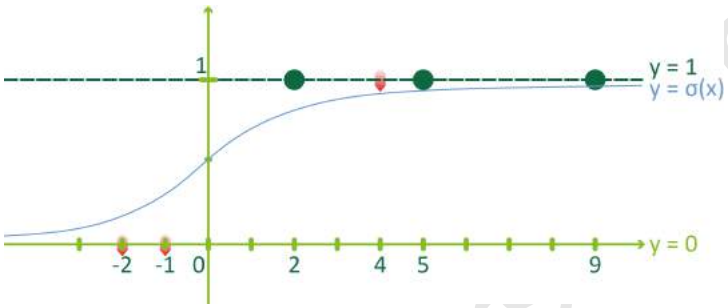
Als $\sigma(x) \geq 0{,}5$, dan wordt het punt ingedeeld bij klasse 1, anders bij klasse 0. $0{,}5$ is de drempelwaarde of *threshold* die wordt gehanteerd.

De netwerk-architect kiest zelf de waarde van de *threshold*, de drempelwaarde, afhankelijk van het soort probleem en de beoogde oplossing.



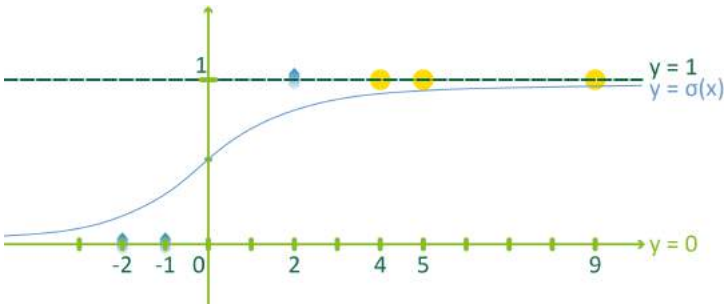
Figuur 10.27: Neuraal netwerk binaire classificatie.

Dit zou enkel Q_4 fout classificeren als horend bij klasse 1, aangezien de zekerheid voor klasse 1 groot is. Dat is een mooi resultaat! Alle punten in de juiste categorie krijgen, gaat immers niet (zie Figuur 10.28).



Figuur 10.28: Eerste sigmoïde om te classificeren in het tweede geval. Er is één punt fout geclassificeerd.

Dezelfde sigmoïde zou geen goed resultaat geven voor het eerste geval, want die klassen zijn perfect scheidbaar en toch zou P_3 fout geclassificeerd worden (zie Figuur 10.29).

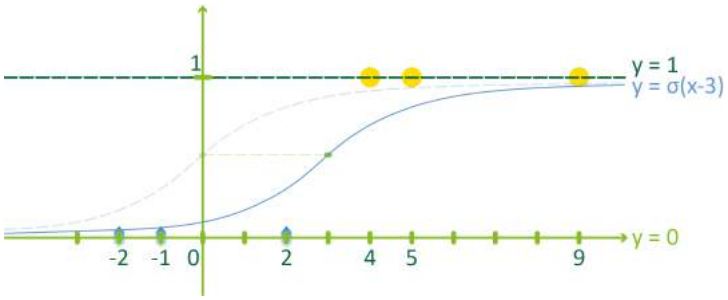


Figuur 10.29: Eerste sigmoïde om te classificeren in het eerste geval. Er is één punt fout geclassificeerd.

In het eerste geval doet men beter het volgende:

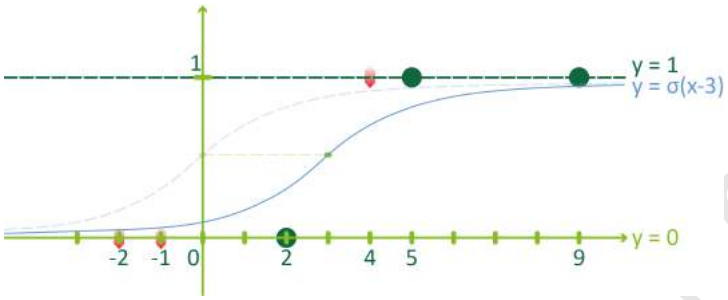
Stel daar $w = -3$, dan: $x \mapsto x - 3 \mapsto \sigma(x - 3)$ (zie Figuur 10.27). Figuur 10.30 toont het resultaat.

Deze sigmoïde zou dan weer een slechtere keuze zijn voor het tweede voorbeeld. Er worden dan immers twee punten bij de foute klasse



Figuur 10.30: Tweede sigmoïde om te classificeren in het eerste geval. Alle punten zijn juist geclassificeerd.

ingedeeld (zie Figuur 10.31).



Figuur 10.31: Tweede sigmoïde om te classificeren in het tweede geval. Er worden twee punten fout geclassificeerd.

Het doel van een ML-model is om te bekomen dat het zo weinig mogelijk punten fout classificeert. Men zegt ook dat de kost zo klein mogelijk moet zijn; men wil de kost minimaliseren.

Notebook - binaire classificatie

- Deze methode van classificatie wordt toegepast in de notebook `StomataZonSchaduw.ipynb` in de map `IntroductieMachineLearning`. De data die gebruikt worden in de notebook, werden ter beschikking gesteld door Miguel Camargo (persoonlijke communicatie via e-mail) (Camargo & Marenco, 2012).
  Deze data zijn bij benadering lineair scheidbaar. Het minimaliseren van de kost gebeurt met *gradient descent* (zie paragraaf 10.11).
- Van 50 bladeren werden de stomatale lengte en de stomatale dichtheid bepaald. Sommige bladeren groeiden in de zon, andere in de schaduw. Zon of schaduw fungeert als label. Het is de bedoeling de bezonde bladeren te scheiden van de beschaduwde.
- Onthoud zeker dat de initiële scheidingsrechte willekeurig werd gekozen. De coëfficiënten in de vergelijking van de rechte spelen de rol van de gewichten die worden aangepast door de gelabelde voorbeelden aan de scheidingsrechte te toetsen.
- Vind je de sigmoïde-functie terug in de code?

10.11 Gradient descent

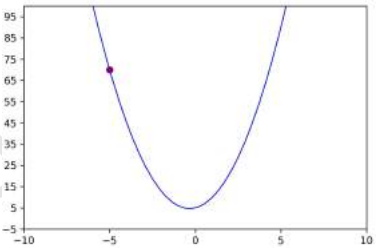
In de wiskunde wordt veel aandacht besteed aan de afgeleide van een reële functie. Afgeleiden hebben veel toepassingen, bv. in de economie en in de fysica. Ook voor neurale netwerken zijn afgeleiden belangrijk. Men gebruikt ze om de kost te minimaliseren.

Notebook - gradient descent

- Gradient descent is gebaseerd op afgeleiden. Beide worden uitgelegd in de notebook `GradientDescent.ipynb` in de map `IntroductieDeepLearning`.
- In de notebook `GradientDescent.ipynb` ga je op zoek naar het minimum van een kwadratische functie van één variabele met gradient descent.

In de notebook `GradientDescent.ipynb` vertrekt men van de parabool met vergelijking $y = 3x^2 + 2x + 5$. Dit is een dalparabool (want $a = 3 > 0$). Deze parabool heeft één minimum.

Er wordt een willekeurig punt P gekozen op de parabool, op de grafiek in Figuur 10.32 afgebeeld in het paars.



Figuur 10.32: Willekeurig punt P op de dalparabool.

Het is de bedoeling het punt P naar het laagste punt van de parabool te laten 'bewegen' en zich daarbij te laten leiden door de afgeleide in het punt P. De x-waarde van P zal dus met bepaalde hoeveelheden moeten worden aangepast, zodat P effectief de top van de parabool nadert. Eens dicht bij de top moet men er alert voor zijn dat men niet voorbij de top gaat.

De learning rate η speelt een belangrijke rol bij gradient descent. Dit getal tussen 0 en 1 bepaalt mee hoe groot de stappen zijn waarmee men de *weights* aanpast. Hoe kleiner de leersnelheid, hoe voorzichtiger de aanpassingen. Een te kleine learning rate kan er echter toe leiden dat het systeem zeer traag leert. Bij een te grote learning rate kan men door te bruuske aanpassingen het minimum gemist hebben.

10.12 Prestatie van het systeem

Eens men een ML-model bekomen heeft, wil men dat gebruiken op nieuwe data. Het model moet dus ook goed presteren op nieuwe data. Men moet dan ook op een of andere manier de betrouwbaarheid van het model kunnen nagaan.

Voordat men het systeem begint te trainen, splitst men daarom de data op in trainingdata, valideringsdata en testdata.

Zowel de **kost** op de trainingdata, valideringsdata als testdata informeren over de prestaties van het systeem.

De **accuraatheid** is voor de prestatie een niet uit het oog te verliezen cijfer. De accuraatheid is het percentage datapunten dat juist wordt geclassificeerd.

Er zijn trouwens nog cijfers die hier informatie kunnen over geven: precision en recall en de F1-score.

Dit model wordt tot slot getest met de testdata. De accuraatheid en de kost bij de testdata worden berekend en vergeleken met die op de trainingdata.

10.13 Stochastic gradient descent

In de voorbeelden die aan bod kwamen, werden de gewichten voor elk datapunt aangepast. In de praktijk wordt meestal *stochastic gradient descent* toegepast.

Bij de opbouw van het netwerk kiest men de grootte van een ***batch***. Bv. voor een trainingset van 60 000 elementen en een batch-grootte 128 zal het systeem dan vóór elke epoch een batch kiezen, het zal m.a.w. 128 elementen willekeurig kiezen uit de trainingset, elk met hun overeenkomstig label. Van elk datapunt in de batch wordt de nodige aanpassing van de gewichten bepaald maar nog niet uitgevoerd; van alle nodige aanpassingen wordt dan het gemiddelde genomen. Deze gemiddelde aanpassing wordt dan ná de epoch uitgevoerd. Bij stochastic gradient descent zal men de nodige aanpassing van de gewichten dus schatten a.d.h.v. een steekproef uit de trainingset. Omdat de dataset doorgaans zeer groot is, bespaart dat veel aan rekenkracht (Goodfellow et al., 2016).

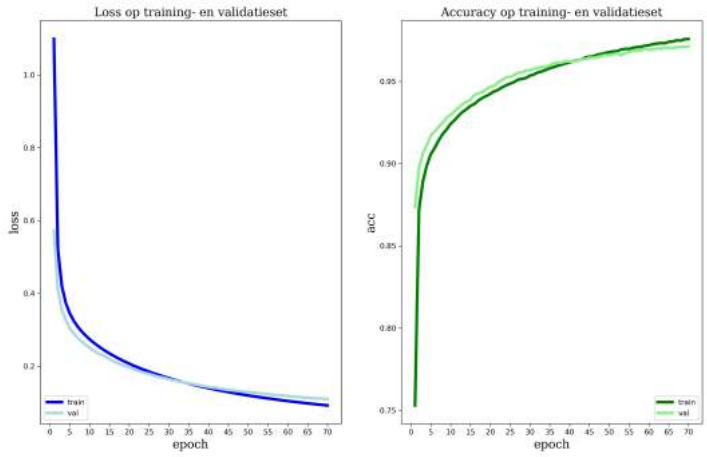
Stochastisch is een synoniem van willekeurig.

10.14 Valideringsdata

Om tijdens het trainen reeds een controle te hebben over hoe het systeem leert, splitst men zelfs de trainingdata nog eens op in de eigenlijke trainingdata en de valideringsdata. De verzameling valideringsdata en de verzameling trainingdata moeten disjunct zijn. Na elke epoch wordt dan nagegaan hoe het systeem generaliseert naar de valideringsdata. Men heeft zo een beter zicht op underfitting en overfitting (zie paragraaf 7.1).

Elk systeem begint op een gegeven moment te overfitten, dan presteert het steeds beter op de trainingdata, maar de prestatie op de validatiedata verbetert niet meer, soms gaat die prestatie er zelfs op achteruit. Het heeft dan geen zin meer verder te gaan met de training. Men streeft immers niet naar een model dat het uitstekend doet op de trainingdata, maar wel naar een model dat zeer goed presteert op ongeziene data.

In de notebooks bij hoofdstuk 11 worden er concrete voorbeelden gegeven van under- en overfitting. De grafieken uit Figuur 10.33 komen uit een van die notebooks (een notebook over de MNIST dataset).

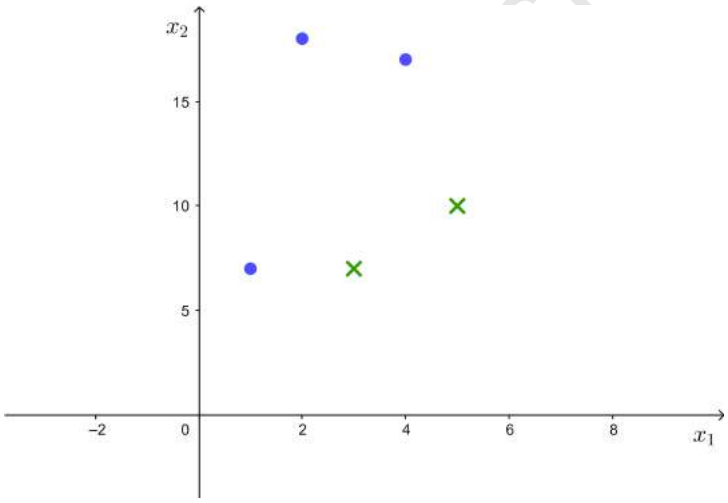


Figuur 10.33: MNIST 70 epochs. Het model overfits.

10.15 *Een model om te classificeren aan het werk en het effect van een niet-lineaire activatiefunctie*

10.15.1 Voorbeeld lineaire scheiding

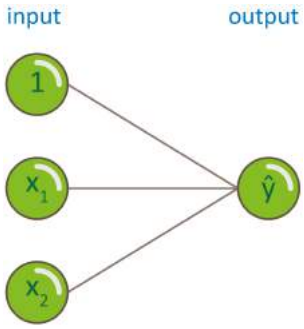
De grafiek in Figuur 10.34 toont punten die voorgesteld worden door blauwe bolletjes en punten die weergegeven worden als groene kruisjes. Deze twee klassen van punten zijn lineair scheidbaar. Stel dat er een neuraal netwerk getraind is en dat er een model beschikbaar is om de twee klassen van elkaar te scheiden.



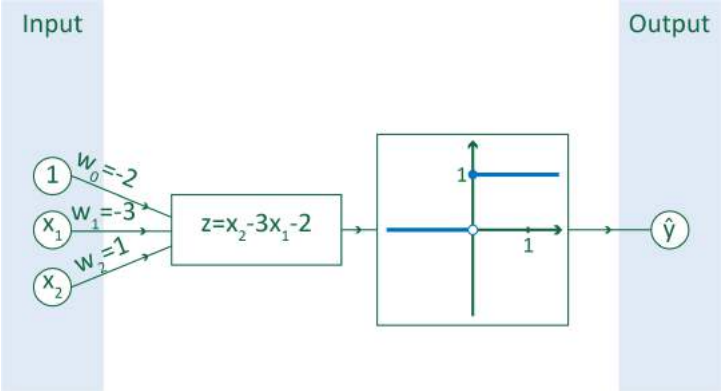
Figuur 10.34: De twee klassen zijn lineair scheidbaar.

Het model wordt geïllustreerd in Figuren 10.36 en 10.35.

Figuur 10.37 toont een voorbeeld van een rechte die de klassen van elkaar scheidt.



Figuur 10.35: Eenvoudige voorstelling van het model voor de lineaire scheiding.



Figuur 10.36: Model voor de lineaire scheiding.

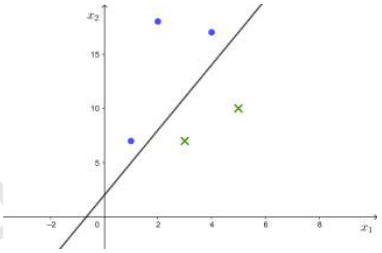
Deze scheidingsrechte heeft als vergelijking

$$x_2 = 3x_1 + 2.$$

De punten die boven deze rechte liggen, voldoen aan $x_2 > 3x_1 + 2$, wat equivalent is met $-3x_1 + x_2 > 2$ en met $-3x_1 + x_2 - 2 > 0$.

De punten die onder deze rechte liggen, voldoen aan $x_2 < 3x_1 + 2$, wat equivalent is met $-3x_1 + x_2 < 2$ en met $-3x_1 + x_2 - 2 < 0$.

Merk het verband op tussen de coëfficiënten in de vergelijking van de scheidingsrechte en de waarden van de gewichten in het model.



Figuur 10.37: Een mogelijke scheidingslijn.

Er wordt dus een lineaire combinatie gemaakt van de inputwaarden x_1, x_2 en een drempelwaarde 2. Vervolgens wordt op de lineaire combinatie de Heaviside-functie toegepast. Bij een positief resultaat wordt er afgebeeld op 1, anders op 0.

De bekomen $\hat{y}$-waarde kan dus 1 of 0 zijn: 1 voor de punten boven de rechte (het gebied van de blauwe bolletjes) en 0 voor de punten eronder (het gebied van de groene kruisjes).

De bias w_0 is -2.

Het voorschrift van deze niet-lineaire functie luidt als volgt:

$$H : \mathbb{R} \to \mathbb{R} : x \mapsto \begin{cases} 0, \text{ als } x < 0 \\ 1, \text{ als } x \geq 0 \end{cases}.$$

Test op het punt $(1,7)$ en het punt $(5,10)$:

$-3 \cdot 1 + 1 \cdot 7 - 2 = 2 > 0$ dus $\hat{y} = 1$, punt boven de rechte (gebied blauwe bolletjes);

$-3 \cdot 5 + 1 \cdot 10 - 2 = -7 < 0$ dus $\hat{y} = 0$, punt onder de rechte (gebied groene kruisjes).

$z = -3 \cdot x_1 + 1 \cdot x_2 - 2.$

In matrixnotatie:

$$X = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}, W = \begin{bmatrix} -2 \\ -3 \\ 1 \end{bmatrix},$$

$$X \mapsto W^T \cdot X = \left[z\right] \mapsto \left[H(z)\right] = \left[\hat{y}\right], \text{ met } \hat{y} \in \{0,1\}.$$

Opdracht - classificatie

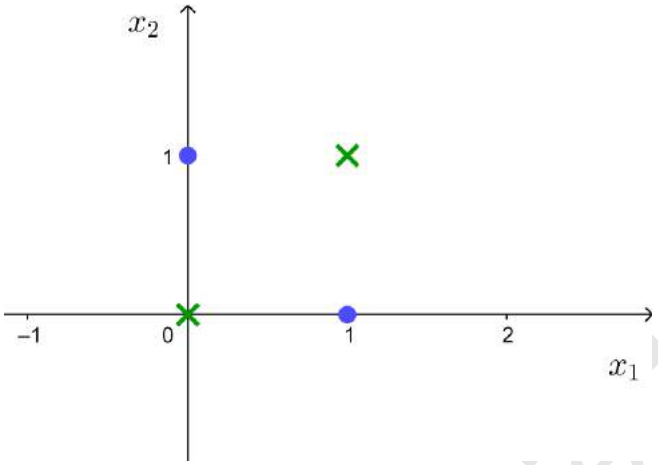
- Test het model uit op het punt (3,7) door gebruik te maken van het schema uit Figuur 10.36.
- Test het model uit op het punt (3,7) door gebruik te maken van de matrixnotatie.

10.15.2 XOR-functie of 'exclusieve of'-functie, één hidden layer

Beschouw de grafiek in Figuur 10.38.

Het voorschrift van de XOR-functie:

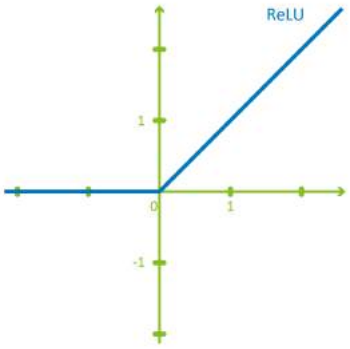
$$XOR: \begin{cases} (0,0) \mapsto 0 \\ (0,1) \mapsto 1 \\ (1,0) \mapsto 1 \\ (1,1) \mapsto 0 \end{cases}.$$



Figuur 10.38: XOR. De punten zijn niet lineair scheidbaar.

Deze grafiek is een voorstelling van de XOR-functie, een booleaanse operator op twee binaire waarden. Als juist één van deze binaire waarden 1 is, dan is het resultaat van de XOR-functie 1, anders is het resultaat 0.

De afgebeelde punten zijn niet lineair scheidbaar. Er is dus geen rechte te vinden die de punten van elkaar scheidt. De punten kunnen toch van elkaar gescheiden worden via een neuraal netwerk dat voor de activatiefunctie gebruikmaakt van de ReLU-functie (zie Figuur 10.39).



Figuur 10.39: ReLU-functie.

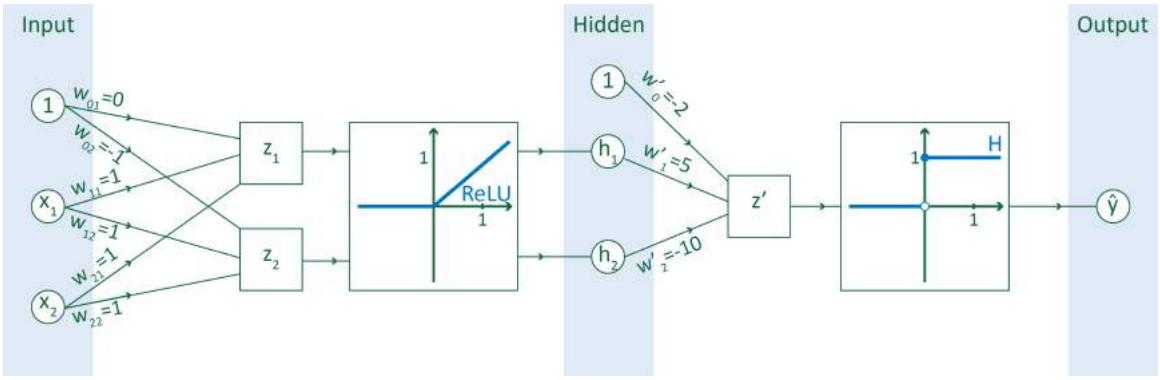
Het voorschrift van de niet-lineaire ReLU-functie luidt als volgt:

$$\text{ReLU}: \mathbb{R} \to \mathbb{R}: x \mapsto \begin{cases} 0, \text{ als } x < 0 \\ x, \text{ als } x \geq 0 \end{cases}.$$

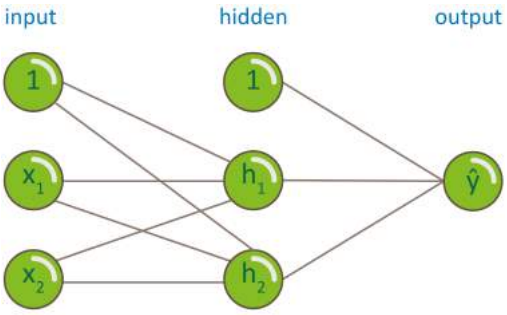
Kort: $\text{ReLU}(x) = max\{0,x\}$.

Figuur 10.40 geeft een voorstelling van een neuraal netwerk dat de punten scheidt. Er worden twee lineaire combinaties gemaakt van de inputwaarden x_1, x_2 en een drempelwaarde. Vervolgens wordt op de lineaire combinaties de ReLU-functie toegepast. Van de bekomen outputwaarden en een drempelwaarde wordt opnieuw een lineaire combinatie gemaakt; hierop wordt de Heaviside-functie toegepast. Bij een positief resultaat wordt de input afgebeeld op $\hat{y} = 1$, anders op $\hat{y} = 0$. Figuur 10.41 toont een vereenvoudigde voorstelling van het model.

In matrixnotatie, na toevoeging van een extra neuron:



Figuur 10.40: Model voor de scheiding.



Figuur 10.41: Eenvoudige voorstelling van het model voor de scheiding.

$$X = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}, W = \begin{bmatrix} 0 & -1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}, W' = \begin{bmatrix} -2 \\ 5 \\ -10 \end{bmatrix}.$$

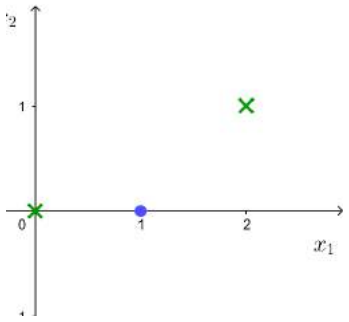
$$X \mapsto W^T \cdot X = Z \mapsto \text{ReLU}(Z) = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix},$$

hieraan wordt ook een extra neuron 1 toegevoegd; men bekomt $H = \begin{bmatrix} 1 \\ h_1 \\ h_2 \end{bmatrix}$ van de *hidden layer*.

$$H \mapsto W'^T \cdot H = \left[z'\right] \mapsto \left[H(z')\right] = \left[\hat{y}\right],$$

met $\hat{y} \in \{0,1\}$.

Grafisch vertaalt zich dat in de grafieken van Figuren 10.42 en 10.43.

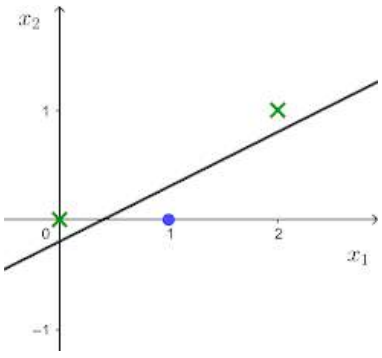


Figuur 10.42: Hidden layer. Lineair scheidbaar.

De scheidingsrechte uit Figuur 10.43 heeft als vergelijking $x_2 = \frac{1}{2}x_1 - \frac{1}{5}$. De punten die boven deze rechte liggen, voldoen aan $x_2 > \frac{1}{2}x_1 - \frac{1}{5}$, wat equivalent is met $\frac{1}{2}x_1 - x_2 < \frac{1}{5}$, met $5x_1 - 10x_2 < 2$ en met $5x_1 - 10x_2 - 2 < 0$. De punten die onder deze rechte liggen, voldoen aan $x_2 < \frac{1}{2}x_1 - \frac{1}{5}$, wat equivalent is met $\frac{1}{2}x_1 - x_2 > \frac{1}{5}$, met $5x_1 - 10x_2 > 2$ en met $5x_1 - 10x_2 - 2 > 0$.

Merk het verband op tussen de coëfficiënten in de vergelijking van deze scheidingsrechte en de waarden van de gewichten W' in het model.

Test het model op het punt $(0,0)$:



Figuur 10.43: Gescheiden door een rechte.

$(1,0,0)$ uit de *input layer* wordt afgebeeld op $(1,0,0)$ in de *hidden layer*; $5 \cdot 0 - 10 \cdot 0 - 2.1 = -2 < 0$ dus $\hat{y} = 0$ (gebied groene kruisjes).

Test het model op het punt $(1,0)$:

$(1,1,0)$ uit de *input layer* wordt afgebeeld op $(1,1,0)$ in de *hidden layer*; $5 \cdot 1 - 10 \cdot 0 - 2.1 = 3 > 0$ dus $\hat{y} = 1$ (gebied blauwe bolletjes).

$z_1 = x_1 + x_2$
$z_2 = x_1 + x_2 - 1$
$z' = 5 \cdot h_1 - 10 \cdot h_2 - 2$

Opdracht - classificatie

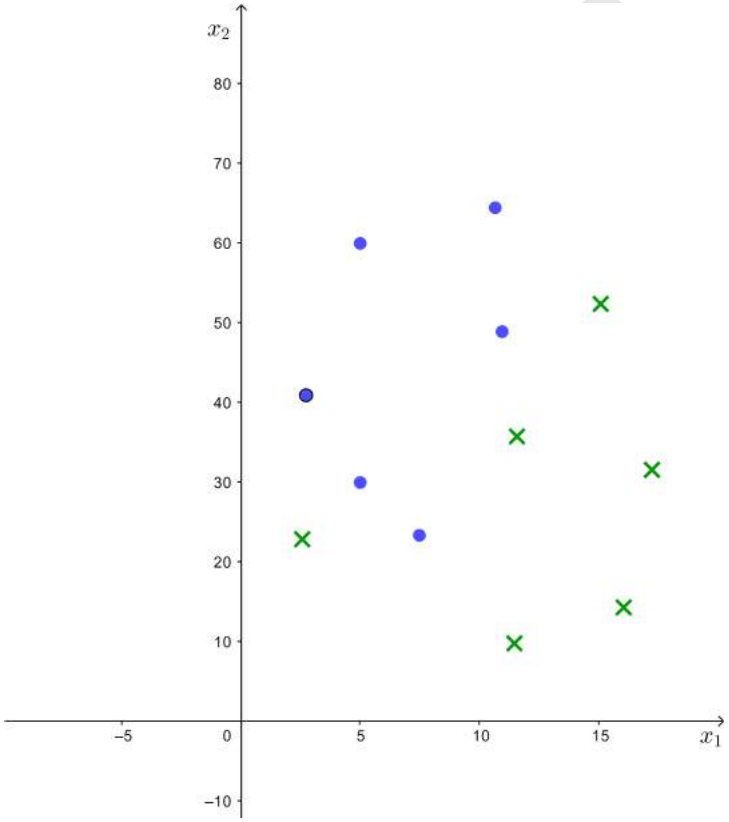
- Test het model uit op de twee andere punten.

ReLU

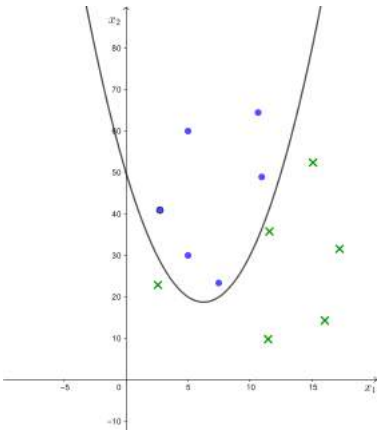
Onthoud: door gebruik te maken van de ReLU-functie, werden de punten lineair scheidbaar. De laag tussen de *input layer* en de *output layer* noemt men een *hidden layer*.

10.15.3 Model met één hidden layer

Beschouw de grafiek in Figuur 10.44.



Figuur 10.44: De klassen zijn niet lineair scheidbaar.



Figuur 10.45: Scheiden met een parabool.

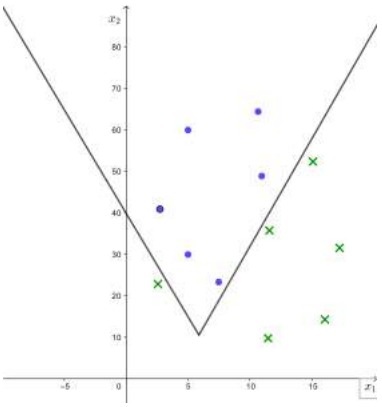
Er zijn twee groepen van punten of anders gezegd twee klassen. Deze punten kunnen van elkaar gescheiden worden, bv. met een

parabool of met twee rechten; zie hiervoor de Figuren 10.45 en 10.46.

De scheiding kan gebeuren met een neuraal netwerk met één hidden layer dat voor de activatiefunctie gebruikmaakt van de sign-functie (zie Figuur 10.47).

Het voorschrift van de niet-lineaire sign-functie luidt als volgt:

$$\text{sign} : \mathbb{R} \to \mathbb{R} : x \mapsto \begin{cases} -1, & \text{als } x < 0 \\ 0, & \text{als } x = 0 \\ 1, & \text{als } x > 0 \end{cases}.$$

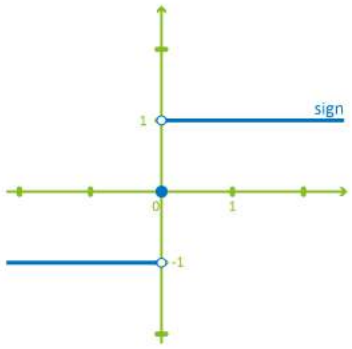


Figuur 10.46: Scheiden met twee halfrechten.

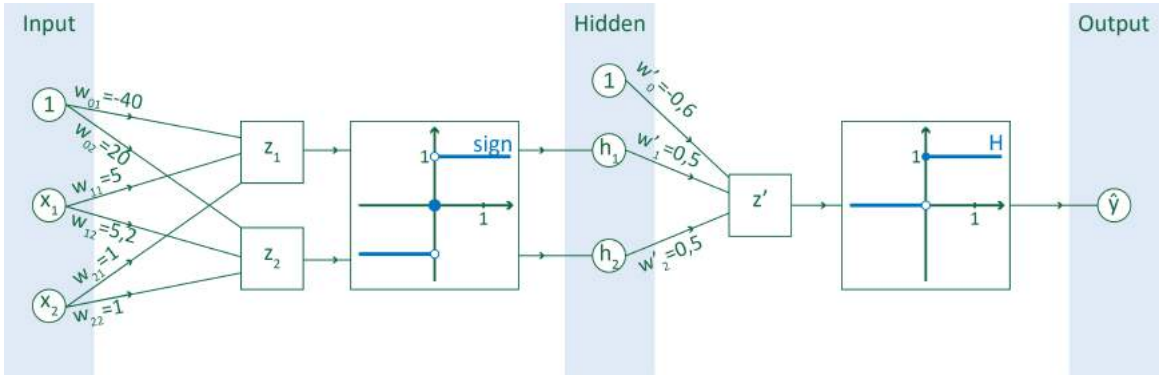
De twee scheidingsrechten in Figuur 10.46 hebben als vergelijking $x_2 = -5\ x_1 + \ 40$ voor de dalende rechte en $x_2 = 5{,}2\ x_1 - 20$ voor de stijgende.

De blauwe punten bv. liggen boven deze twee halfrechten en voldoen aan $\begin{cases} x_2 > -5\ x_1 + 40 \\ x_2 > 5{,}2\ x_1 - 20 \end{cases}$, wat equivalent is met $\begin{cases} 5\ x_1 + x_2 > 40 \\ -5{,}2\ x_1 + x2 > -20 \end{cases}$ en met $\begin{cases} 5\ x_1 + x_2 - 40 > 0 \\ -5{,}2\ x_1 + x2 + 20 > 0. \end{cases}$

Een voorstelling van het model staat in Figuren 10.48 en 10.49. Er worden dus twee lineaire combinaties gemaakt van de inputwaarden x_1 en x_2, een met drempelwaarde 40 en een met drempelwaarde -20. Vervolgens wordt op de lineaire combinaties de sign-functie toegepast om de waarden in de hidden layer te bekomen. Er wordt dan een lineaire combinatie gemaakt van deze waarden en een drempelwaarde, waarop de Heaviside-functie wordt toegepast. Bij een positief resultaat wordt de input afgebeeld op $\hat{y} = 1$, anders op $\hat{y} = 0$.



Figuur 10.47: Sign-functie.



Figuur 10.48: Neuraal netwerk dat de punten scheidt, met activatiefuncties sign en Heaviside.

De outputwaarde $\hat{y}$ kan twee waarden aannemen: 0 voor het gebied van de groene kruisjes en 1 voor het gebied van de blauwe bolletjes.

In matrixnotatie:

$$X = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}, W = \begin{bmatrix} -40 & 20 \\ 5 & -5{,}2 \\ 1 & 1 \end{bmatrix}, W' = \begin{bmatrix} -0{,}6 \\ 0{,}5 \\ 0{,}5 \end{bmatrix}.$$

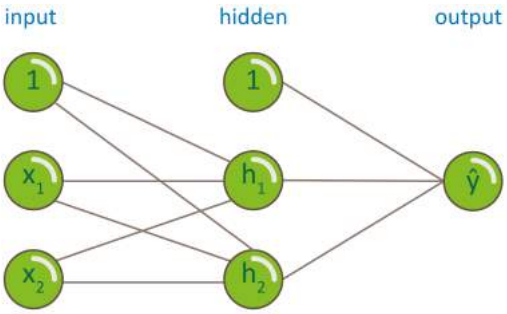
$$X \mapsto W^T \cdot X = Z \mapsto \text{sign}(Z) = \begin{bmatrix} h_1 \\ h_2 \end{bmatrix},$$

hieraan wordt een extra neuron toegevoegd,

$$H = \begin{bmatrix} 1 \\ h_1 \\ h_2 \end{bmatrix}.$$

$$H \mapsto W'^T \cdot H = \left[z'\right] \mapsto \left[H(z')\right] = \left[\hat{y}\right],$$

met $\hat{y} \in \{0{,}1\}$.



Figuur 10.49: Vereenvoudigde voorstelling van het model voor de scheiding.

Test op de punten $(0{,}0)$ en $(5, 30)$:

$5 \cdot 0 + 1 \cdot 0 - 40 = -40 < 0$ en $-5{,}2 \cdot 0 + 1 \cdot 0 + 20 = 20 > 0$,

dus $H = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$, dus $\hat{y} = 0$ (gebied groene kruisjes).

$z_1 = 5 \cdot x_1 + 1 \cdot x_2 - 40$
$z_2 = -5{,}2 \cdot x_1 + 1 \cdot x_2 + 20$
$z' = 0{,}5 \cdot h_1 + 0{,}5 \cdot h_2 - 0{,}6$

$5 \cdot 5 + 1 \cdot 30 - 40 = 15 > 0$ en $-5{,}2 \cdot 5 + 1 \cdot 30 + 20 = 24 > 0$,

dus $H = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$, dus $\hat{y} = 1$ (gebied blauwe bolletjes).

Opdracht - classificatie

- Test het model uit op het punt (15,48).
- Test het model uit op nog een ander punt dat je zelf kiest.
- Gebruik zowel het schema als de berekening met matrices.

Onthoud: y en $\hat{y}$ hebben evenveel mogelijke waarden als er klassen zijn.

In de netwerken, met een verborgen laag, uit de Figuren 10.40, 10.41, 10.48 en 10.49 zijn alle neuronen van de opeenvolgende lagen met elkaar verbonden; men spreekt van ***dense layers*** of *fully connected layers*.

Notebook - opdracht niet-lineaire classificatie

- Doorloop de notebook `Relu.ipynb` in de map `IntroductieDeepLearning` en bekijk hoe ReLU helpt om niet-lineair scheidbare data toch lineair scheidbaar te maken.
- Maak ook de opdracht in deze notebook waarbij je de grens tussen de klassen van niet-lineair scheidbare data visualiseert en zelf aan de slag gaat met ReLU.
- Bouw voor deze opdracht zelf een schema op dat het model voorstelt.

Samengevat - computerwetenschappen

Het Perceptron, het eerste neurale netwerk, heeft enkel een invoer- en een uitvoerlaag. Het Perceptron is geschikt om lineair scheidbare klassen van elkaar te scheiden.

Als klassen niet lineair scheidbaar zijn, dan zal het Perceptron niet volstaan. Daarvoor zijn er andere, meer complexe, neurale netwerken. Deze netwerken hebben meer lagen en gebruiken ook andere activatiefuncties, zoals ReLU, de sigmoïde-functie en de sign-functie.
Bij elk neuraal netwerk streeft men ernaar om de fout zo klein mogelijk te maken. Men zal de kostenfunctie minimaliseren met *(stochastic) gradient descent*.
Voor een binaire classificatie gebruikt men bv. een netwerk met één verborgen laag met de sigmoïde-functie als activatiefunctie; *binary cross-entropy*, BCE, is de kostenfunctie.

Behalve classificatie kan men met neurale netwerken ook regressieproblemen aanpakken, bv. lineaire regressie, waarbij men op zoek gaat naar de best passende rechte bij de data. Bij lineaire regressie is er eigenlijk geen activatiefunctie nodig; men zegt dat de identiteit de activatiefunctie is. De gemiddelde kwadratische afwijking, *mean squared error*, MSE, doet er dienst als kostenfunctie.

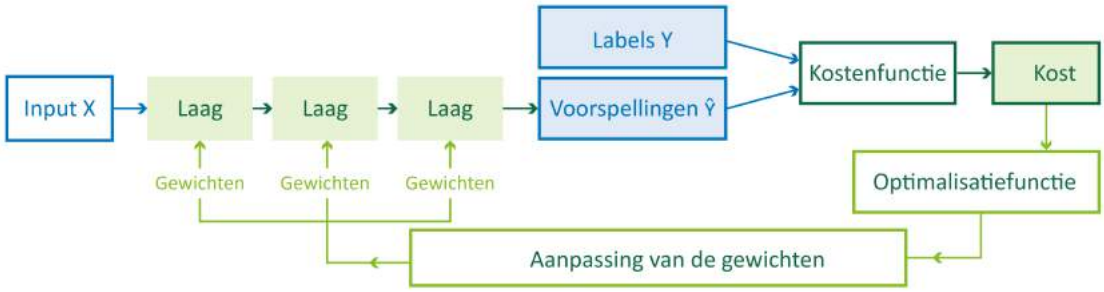
Voordat men een netwerk traint, splitst men de data op in trainingdata, valideringsdata en testdata. De kostenfunctie op deze data, maar ook de accuraatheid zullen informeren over de prestaties van het model.
Bij het trainen moet men beducht zijn voor *underfitting* (de prestaties van het systeem kunnen bv. door het kiezen van een ander model of door meer te trainen nog verbeteren) en *overfitting* (het systeem presteert steeds beter op de trainingdata, maar de prestaties op de valideringsdata verbeteren niet meer).

HOOFDSTUK
ELF

IMPLEMENTATIE IN KERAS

11.1 Diepe neurale netwerken van KIKS

Een diep lerend netwerk bestaat uit meerdere lagen die aaneengeschakeld zijn. Elke laag beschikt over een aantal parameters, zijn gewichten. Figuur 11.1 stelt de opbouw van een diep lerend netwerk schematisch voor (Chollet, 2018).



Figuur 11.1: Opbouw van een diep neuraal netwerk.

De input X (uit de trainingdata) wordt gegeven aan de eerste laag, de input layer. Via berekeningen binnen deze layer wordt een output bekomen; X is getransformeerd naar een nieuwe gedaante.

Deze output gaat dan naar de volgende layer, waar een soortgelijk proces plaatsvindt.

Tot slot komt de output van de voorlaatste layer aan in de laatste layer, de output layer. De output layer heeft ook een output: het systeem doet een voorspelling $\hat{y}$.

Deze voorspelling wordt vergeleken met het gegeven label. Daarvoor is de loss functie nodig. De mate waarin de voorspelling verschilt van het label, bepaalt de aanpassingen die gedaan worden aan de gewichten. Daarvoor is dan weer de optimizer nodig.

Een eerste epoch is gedaan. Het systeem onderneemt een tweede epoch: het proces begint opnieuw, maar nu met de aangepaste gewichten.

Het aanpassen van de gewichten is wat men bedoelt met dat

het systeem leert. De 'kennis' van het netwerk is opgeslagen in de gewichten.

De Python-module Keras voorziet bouwblokken om op een vrij eenvoudige manier een neuraal netwerk op te bouwen.

Dankzij Keras wordt heel wat werk bespaard: in de achterliggende code zijn de nodige functionaliteiten vervat. Het oplossen van bepaalde problemen met een deep learning netwerk ligt binnen ieders bereik, mits er voldoende rekenkracht voorhanden is.
Voor het rekenen met tensoren en voor andere rekenkundige bewerkingen doet Keras zelf een beroep op de module TensorFlow.

```
Using TensorFlow backend.
```

Figuur 11.2: Keras refereert aan TensorFlow.

De netwerkarchitect moet wel nog **keuzes** maken en moet de data in de juiste vorm aan het netwerk aanbieden:

- het type probleem bepalen, bv. regressie, binaire of multiklassenclassificatie;
- het type neuraal netwerk bepalen;
- de lagen kunnen gewone lagen zijn of convolutionele lagen;
- de activatiefunctie van elke laag;
- het aantal lagen en gewichten;
- vóór de training de data opsplitsen in trainingdata, valideringsdata en testdata;
- loss, optimizer en metrics;
- het aantal epochs;
- de grootte van eventuele batches;
- enkele hyperparameters kiezen: de *learning rate* en eventueel een *threshold*.

11.2 Netwerkarchitectuur van de KIKS-notebooks

De keuzes die de netwerkarchitect maakt, worden bepaald door het probleem, maar grotendeels ook door zijn ervaring. Hij houdt bv. rekening met het volgende:

- Meer lagen of meer gewichten hebben als voordeel dat er meer patronen ontdekt kunnen worden, wat de prestatie van het netwerk verhoogt. Bij een classificatieprobleem met n klassen vermijdt men het best hidden layers met veel minder dan n weights. Dat zou er immers voor zorgen dat het systeem niet alle relevante informatie kan leren.
- Voor problemen rond bv. computer vision heeft men met een convolutioneel netwerk meer kans op betere resultaten dan zonder convoluties. Met minder epochs kan dan een beter resultaat bekomen worden. Let wel: elke epoch kost meer rekenkracht en zal langer duren.

In de tabellen 11.1, 11.2 en 11.3 worden enkele vuistregels meegegeven.

	Tips regressie
y, $\hat{y}$	continue waarden
Voorspellingen naar de toekomst	testset bestaat uit de 'laatste' waarden
Probleem zonder tijdsaspect	testset kan hier willekeurig uit de dataset worden gekozen
Standaardiseren	steeds de data standaardiseren
Loss	mean squared error (MSE)
Optimizer	gradient descent (GD) of stochastic gradient descent (SGD)
Tussenlagen	activatiefunctie: ReLU
Output layer	geen activatiefunctie
Metrics	meerdere mogelijkheden, bv. mean absolute error (MAE)

Tabel 11.1: Tips bij regressieprobleem.

	Tips binaire classificatie
y, $\hat{y}$	$y \in \{0,1\}$, $\hat{y} \in \{0,1\}$
Twee klassen	een klasse krijgt label 1 en de andere krijgt label 0
Testset	testset kan willekeurig uit de dataset worden gekozen
Standaardiseren/normaliseren	de data standaardiseren of normaliseren
Loss	binary cross-entropy
Optimizer	gradient descent (GD) of stochastic gradient descent (SGD)
Tussenlagen	activatiefunctie: ReLU
Output layer	activatiefunctie: sigmoïde deze geeft hoe zeker het model is dat gegeven tot klasse 1 behoort om de klasse effectief te voorspellen moet er een threshold worden gekozen, bv. vanaf 75 % zekerheid, voorspel klasse 1
Metrics	accuracy

Tabel 11.2: Tips bij binair classificatieprobleem.

In de KIKS-notebooks worden neurale netwerken opgebouwd. Bij KIKS gebruiken we een model dat bestaat uit aaneengeschakelde lagen, het zogenaamde ***Sequential model***.

Het model moet weten welke vorm van input het kan verwachten, m.a.w. wat de **dimensie** is van de **inputpunten**. Daarom wordt dit aan de eerste laag van het Sequential model meegegeven. Dat moet enkel voor de eerste laag, want de volgende lagen verkrijgen dat automatisch, aangezien die dimensie bepaald wordt door de wiskundige bewerkingen die er gebeuren.

Als de learning rate niet expliciet vermeld staat in de netwerkopbouw van het Pythonscript, dan gebruikt het netwerk een reeds vooropgestelde learning rate, bv. $0{,}01$ bij *stochastic gradient descent* (SGD).

Vóór er dan effectief met trainen kan begonnen worden, moeten de gegevens eventueel nog omgevormd worden naar het juiste **formaat**: tensoren met de juiste dimensie, kwantitatieve labels i.p.v. kwalitatieve, *one hot encoding* i.p.v. $0, 1, 2, 3$ ...

Het **trainen** kan dan gebeuren. Hiervoor wordt de **methode `fit()`** gebruikt. De kenmerken en de labels worden op elkaar afgestemd. Tijdens de training wordt na elke epoch de loss en accuracy op de valideringsset getoond. Zolang de loss daalt en de accuracy stijgt, verloopt de training naar wens.

De methode `fit()` geeft een object terug van de klasse `History`. Dit object heeft een attribuut `history` waarin de waarden van de training loss-waarden en de *metrics*-waarden van de opeenvolgende epochs zijn bewaard en, indien van toepassing, ook de loss en de metrics van de valideringsset.

	Tips multiklassenclassificatie
y, $\hat{y}$	bv. bij MNIST $y \in \{0,1,2,...,9\}$, $\hat{y} \in \{0,1,2,...,9\}$
n klassen	als de klassen een kwalitatieve waarde hebben, dan wordt die waarde omgezet naar een kwantitatieve waarde, meer bepaald een discrete waarde bv. bij MNIST: 0 t.e.m. 9, soort iris wordt 0, 1 en 2 deze discrete waarden worden nog eens omgezet via *one hot encoding*: bv. eerste klasse is bv. 100, de tweede klasse 010 en de derde klasse 001
Testset	testset kan willekeurig uit de dataset worden gekozen
Standaardiseren/normaliseren	de data standaardiseren of normaliseren
Loss	categorical cross-entropy
Optimizer	gradient descent (GD) of stochastic gradient descent (SGD)
Tussenlagen	activatiefunctie: ReLU
Output layer	activatiefunctie: softmax deze geeft hoe zeker het model is dat gegeven tot een van de klassen behoort, dus geeft voor elke klasse de zekerheid
Metrics	accuracy

Tabel 11.3: Tips bij problemen met meer dan twee klassen.

11.3 *Implementatie in Keras*

Notebook

Er zijn meerdere KIKS-notebooks waarin met Keras een neuraal netwerk wordt geïmplementeerd. Je vindt ze in de map `IntroductieDeepLearning`.

- `MNIST.ipynb`
- Binaire classificatie met neuraal netwerk met een verborgen laag
- Binaire classificatie met neuraal netwerk met twee verborgen lagen
- Multiklassenclassificatie voor twee klassen met neuraal netwerk met twee verborgen lagen
- Classificatie van de MNIST dataset met een convolutioneel neuraal netwerk
- Classificatie van de Iris dataset met een verborgen laag
- Classificatie van de Iris dataset met alle kenmerken
- Binaire classificatie met neuraal netwerk met een verborgen laag en met verbeterde SGD

Samengevat - computerwetenschappen

Een diep lerend netwerk bestaat uit meerdere lagen die aaneengeschakeld zijn. Deze netwerken hebben een simpele opbouw die verwezenlijkt kan worden met de bouwblokken van de Python-module Keras.
De netwerken van KIKS gebruiken het *Sequential model* van Keras.
De architect van het netwerk moet de data voorbereiden. Bij de opbouw van het netwerk maakt de netwerkarchitect heel wat keuzes.

BIBLIOGRAFIE

American Museum of Natural History (2014). *Keeling's Curve. The Story of CO2* [Videobestand]. Geraadpleegd op 9 augustus 2020 via https://youtu.be/0Z8g-smE2sk.

Assouline, S. & Or, D. (2013). Plant Water Use Efficiency over Geological Time: Evolution of Leaf Stomata Configurations Affecting Plant Gas Exchange. *PloS one*, *8.e67757*.

Bauters, M., Meeus, S., Barthel, M., Stoffelen, P., Deurwaerder, H. D., Meunier, F., Drake, T. W., Ponette, Q., Ebuy, J., Vermeir, P., Beeckman, H., wyffels, F., Bodé, S., Verbeeck, H., Vandelook, F., & Boeckx, P. (2020). Century-long apparent decrease in iWUE with no evidence of progressive nutrient limitation in African tropical forests. *Global Change Biology*, *26*(8), 4449–4461.

Bergmann, D. (2015). *Stomata and global climate cycles* [Videobestand]. Geraadpleegd op 30 augustus 2019 via https://youtu.be/eD2J3PBoERI.

Binek, R. (2019). *Kosaciec szczecinkowaty Iris setosa* [Afbeelding]. CC BY-SA 3.0. Verkregen via Wikimedia Commons https://commons.wikimedia.org/wiki/File:Kosaciec_szczecinkowaty_Iris_setosa.jpg.

Boden, M. A. (2016). *AI. Its nature and future*. Oxford, Verenigd Koninkrijk: Oxford University Press.

Boussemaere, P. (2015). *Eerste hulp bij klimaatverwarring. Waarom de opwarming van de aarde veel meer is dan een milieuprobleem*. Leuven: Davidsfonds Uitgeverij nv.

Bromet, F. (2016). *Mobiel bellen in 1998* [Videobestand]. Geraadpleegd op 8 augustus 2020 via https://youtu.be/TNwhIHqM60g.

Camargo, M. & Marenco, R. (2012). Growth, leaf and stomatal traits of crabwood (*Carapa guianensis* Aubl.) in Central Amazonia. *Revista Árvore*, *36*, 07–16.

CarbonTracker (2019). *Carbon Dioxide Pumphandle 2019* [Videobestand]. Geraadpleegd op 9 augustus 2020 via https://www.esrl.noaa.gov/gmd/ccgg/trends/history.html.

Chollet, F. (2018). *Deep learning with Python*. Shelter Island, NY: Manning Publications Co.

CSIRO. Atmospheric Research (2019). *Bubbles in Ice* [Afbeelding]. CC BY 3.0. Geraadpleegd op 23 september via https://www.scienceimage.csiro.au/image/521/bubbles-in-ice/.

davcjal (2015). *Stomatal Closure in Tradescantia Leaf Cells* [Videobestand]. Geraadpleegd op 30 augustus 2019 via https://youtu.be/AwyrqfNTuxQ.

de Boer, H. J., Price, C., Wagner-Cremer, F., Dekker, S., Franks, P., & Veneklaas, E. (2016). Optimal allocation of leaf epidermal area for gas exchange. *New Phytologist*, *210*(4), 1219–1228.

Dua, D. & Taniskidou, E. K. (2017). *UCI Machine Learning Repository.* Irvine, CA: University of California, School of Information and Computer Science. Geraadpleegd op 28 juli 2019 via http://archive.ics.uci.edu/ml.

DWDD (2019). *Mobiel bellen: 1998 vs NU* [Videobestand]. Geraadpleegd op 8 augustus 2020 via youtu.be/mQVkpdzPGtQ.

Eliza (2018). *ELIZA conversation* [Afbeelding]. Publiek domein. Geraadpleegd op 28 juli 2019 via https://commons.wikimedia.org/wiki/File:ELIZA_conversation.jpg.

Fischer, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, *7*(2), 179–188.

Franks, P. J. & Beerling, D. J. (2009). Maximum leaf conductance driven by CO_2 effects on stomatal size and density over geologic time. *Proceedings of the National Academy of Sciences*, *106*(25), 10343–10347.

Futureism (2017). *This 'Racist soap dispenser' at Facebook office does not work for black people* [Videobestand]. Geraadpleegd op 8 augustus 2020 via https://youtu.be/YJjv_0eiHmo.

Godoy, D. (2018). Understanding binary cross-entropy / log loss: a visual explanation [Blogpost]. Geraadpleegd op 11 augustus 2019 via https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning.* Cambridge, MA: MIT Press.

Güzeldere, G. & Franchi, S. (1995). Dialogues with colorful "personalities" of early AI. *Stanford Hum. Rev.*, *4*(2), 161–169.

Haykin, S. (2009). *Neural Networks and Learning Machines.* New Jersey: Pearson Education.

Hetherington, A. M. & Woodward, F. I. (2003). The role of stomata in sensing and driving environmental change. *Nature*, *424*, 901–908.

IPCC (2018). IPCC: Summary for Policymakers. In *Global Warming of 1.5℃: An IPCC Special Report on the impacts of global warming of 1.5℃ above pre-industrial levels and related global greenhouse gas emission pathways, in the context of strengthening the global response to the threat of climate change, sustainable development, and efforts to eradicate poverty*. [V. Masson-Delmotte and P. Zhai and H.-O. Pörtner and D. Roberts and J. Skea and P.R. Shukla and A. Pirani and W. Moufouma-Okia and C. Péan and R. Pidcock and S. Connors and J.B.R. Matthews and Y. Chen and X. Zhou and M.I. Gomis and E. Lonnoy and T. Maycock and M. Tignor and T. Waterfield (eds.)]. World Meteorological Organization, Geneva, Switzerland, 32 pp. Geraadpleegd op 5 september 2019 via https://www.ipcc.ch/sr15/chapter/spm/.

Langlois, D. (2019). No machine-readable author provided. dlanglois assumed (based on copyright claims). *Iris* [Afbeelding]. CC BY-SA. Verkregen via Wikimedia Commons https://commons.wikimedia.org/wiki/File:Iris_versicolor_3.jpg.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*, 436–444.

Lim, M. (2018). History of AI Winters. Geraadpleegd op 27 december 2018 via https://www.actuaries.digital/2018/09/05/history-of-ai-winters/.

Maslin, M. (2014). *Climate Change: A Very Short Story.* Oxford, Verenigd Koninkrijk: Oxford University Press.

Mayfield, F. (2007). *Iris virginica shrevei BLUE FLAG* [Afbeelding]. CC BY-SA 2.0, verkregen via Wikimedia Commons https://commons.wikimedia.org/wiki/File:Iris_virginica.jpg.

McGrath, S. (2007). *Bamboo* [Afbeelding]. CC BY 2.0, via Wikimedia Commons. Geraadpleegd op 19 mei 2019 via https://nl.wikipedia.org/wiki/Bestand:Bamboo_(1466706101).jpg.

Nardone, C. (2007). *Shakey the Robot* [Afbeelding]. CC BY-SA 2.0. Geraadpleegd op 28 juli 2019 via https://commons.wikimedia.org/wiki/File:Shakey_the_Robot_(developed_between_1966-1972_at_SRI_International)_-_Computer_History_Museum_(2007-11-10_23.16.01_by_Carlo_Nardone).jpg.

NASA (2012). *The Moon and Earth's atmosphere* [Afbeelding]. Geraadpleegd op 24 oktober 2019 via https://www.nasa.gov/mission_pages/station/multimedia/gallery/iss030e031275.html.

NASA (2020). Sea level. Geraadpleegd op 20 augustus 2020 via https://climate.nasa.gov/vital-signs/sea-level/.

NCEI (2019). What are proxy data? Geraadpleegd op 29 mei 2019 via https://www.ncdc.noaa.gov/news/what-are-proxy-data.

Nerbonne, J. (2004). Logistische regressie. Geraadpleegd op 15 augustus 2020 via https://www.let.rug.nl/nerbonne/teach/stats/Moore-McCabe-H15.pdf.

NOAA (2008). Temperature Change and Carbon Dioxide Change. Geraadpleegd op 31 augustus 2019 via https://www.ncdc.noaa.gov/global-warming/temperature-change.

Rijksuniversiteit Groningen (2005). *Neurale Netwerken* [PowerPoint slides]. Geraadpleegd op 20 oktober 2019 via https://www.slideserve.com/yitro/neurale-netwerken.

Robinson, R. (2017). Convolutional Neural Networks - Basics: An Introduction to CNNs and Deep Learning. Geraadpleegd op 10 oktober 2019 via https://mlnotebook.github.io/post/CNN1/.

Rupp, K. (2019). *Data transistoren*. Geraadpleegd op 6 september 2019 via https://github.com/karlrupp/microprocessor-trend-data.

Sci- Inspi (2018). *Monocot and Eudicot Germination Time-lapse* [Videobestand]. Geraadpleegd op 30 augustus 2019 via https://youtu.be/WbG5zu2Vw0I.

Scripps Institution of Oceanography (2019). *The Keeling Curve*. Geraadpleegd op 30 augustus 2019 via https://scripps.ucsd.edu/programs/keelingcurve/.

Steels, L., Berendt, B., Pizurica, A., & Vandewalle, J. (2017). *Artificiële intelligentie. Naar een vierde industriële revolutie?* Brussel: KVAB Standpunt 53.

Steinthorsdottir, M., Vajda, V., & Pole, M. (2019). Significant transient pCO_2 perturbation at the New Zealand Oligocene-Miocene transition recorded by fossil plant stomata. *Palaeogeography, Palaeoclimatology, Palaeoecology*, *515*, 152–161.

Sterrenwacht Armand Pien (2018). *Fotosynthese*. Geraadpleegd op 24 december 2018 via https://www.rug-a-pien.be/docs/fotosynthese.pdf.

Tans, P. (2018). National Oceanic and Atmospheric Administration (NOAA), Verenigde Staten. Via e-mail.

Tegmark, M. (2017). *LIFE 3.0*. New York, Verenigde Staten: Vintage Books, Penguin Random House, eerste editie edition.

Thanukos, A. (2018). Ancient fossils and modern climate change: The work of Jennifer McElwain, based upon interviews with Jennifer McElwain. https://evolution.berkeley.edu/evolibrary/article/mcelwain_01 t.e.m. https://evolution.berkeley.edu/evolibrary/article/mcelwain_08.

Vlaamse Milieumaatschappij (2019). *Het milieurapport Vlaanderen*. Geraadpleegd op 29 mei 2019 via https://www.milieurapport.be/milieuthemas/klimaatverandering.

Yamori, W., Hikosaka, K., & Way, D. A. (2013). Temperature response of photosynthesis in C3, C4, and CAM plants: temperature acclimation and temperature adaptation. *Photosynthesis research*, *119*.

Zephyris (2011). *Leaf Tissue Structure* [Afbeelding]. Aangepast. CC BY-SA 3.0. Geraadpleegd op 3 maart 2019 via https://commons.wikimedia.org/w/index.php?title=File:Leaf_Tissue_Structure.svg&oldid=333913566.

KIKS is een STEM-project rond artificiële intelligentie voor de derde graad secundair onderwijs. Leerlingen leren AI begrijpen, met mogelijkheden en beperkingen; ze leren hoe ze er een impact kunnen op hebben.

De relatie tussen huidmondjes (stomata) van planten en de klimaatverandering biedt een uniek kader om met diepe neurale netwerken aan de slag te gaan. De programmeertaal Python is bovendien zeer toegankelijk als tool om de fundamenten van neurale netwerken te bestuderen.

Een troef van het KIKS-project is de intensieve samenwerking tussen onderzoekers en leerkrachten. Het lesmateriaal van KIKS wordt immers ontwikkeld, parallel met de resultaten van een lopend wetenschappelijk onderzoek aan de UGent en de Plantentuin Meise.

9 789081 991797

KIKS