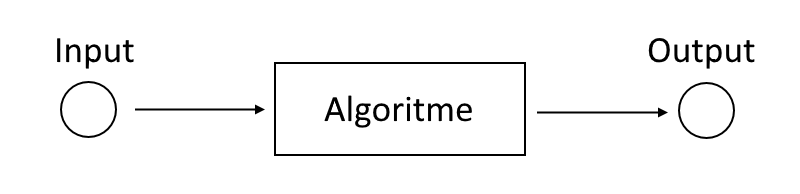
Wat zijn zelflerende systemen?

# Machine learning

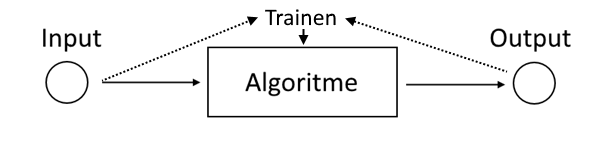
Een zelflerend systeem is een algoritme gebaseerd op machine learning. Machine learning werd door Arthur Samuel, een pionier op dit gebied, gedefinieerd als: "A field of study that gives computers the ability to learn without being explicitly programmed.” [1]. In tegenstelling tot de eerder genoemde algoritmes is een zelflerend systeem in staat zichzelf te verbeteren. Hierdoor kan het taken uitvoeren waarbij reguliere algoritmes tekort schieten. Welke taken dit betreft, zullen we in de derde deelvraag behandelen.

Figuur Zelflerend systeem systematisch

In figuur \*\*\*\* is een schematische weergave van een zelflerend systeem afgebeeld. Bepaalde input data gaat het systeem in en bepaalde output data komt het systeem uit. De input en output data bestaat uit één of meerdere getallen. Als de input simpelweg een reeks getallen betreft, zal dit direct als input gebruikt kunnen worden. In het geval dat de input uit een ander datatype bestaat, zoals een plaatje, zal dit omgezet moeten worden in een reeks getallen voordat het in een zelflerend systeem gebruikt kan worden. Het algoritme zal deze getallen bewerken tot de gewenste output. Deze output wordt eveneens in getallen gegeven. Waar nodig zullen deze getallen dus weer moeten worden omgezet tot het gewenste datatype.

Er zijn vele manieren waarop het systeem zichzelf kan verbeteren. Er zijn veel verschillende algoritmes die gebruikt kunnen worden voor een zelflerend systeem. Elk algoritme heeft voor- en nadelen en is geschikt voor andere doeleinden. Een aantal van deze algoritmes zullen we in de tweede deelvraag behandelen.

Training

Een zelflerend systeem begint in de meeste gevallen zonder enige kennis van de data. Om de gewenste output te kunnen produceren is het dus nodig om het systeem eerst input data te geven zodat het kan leren. Dit proces wordt het “trainen” genoemd. Voor het trainen van een zelflerend systeem is training data nodig. Deze data moet gelijk of gelijkwaardig zijn aan de “echte” data. De training data kan in veel verschillende vormen voorkomen en de manier van trainen is afhankelijk van de vorm van de (training) data. In figuur \*\*\*\* is te zien dat het trainen los staat van het algoritme. Dit verschil zullen we in de volgende deelvraag wat duidelijker maken.   
Er zijn drie prominente manieren waarop een zelflerend systeem getraind kan worden: **supervised**, **unsupervised** en **reinforcement learning**.

Supervised Learning

Figuur Schematische weergave van een zelflerend systeem

In het geval van supervised learning heb je te maken met ***labeled training data***. Anders gezegd: van een bepaalde input is de gewenste output al bekend. Een klassiek voorbeeld van een labeled dataset is een dataset van huisprijzen en huiseigenschappen (zie figuur xx)

|  |  |  |  |
| --- | --- | --- | --- |
| Huisprijs (output) | Huiseigenschappen (input) | | |
|  | Woonoppervlakte | Perceeloppervlakte | Aantal kamers |
| € 519.000 | 124 m² | 311 m² | 4 |
| € 569.000 | 133 m² | 309 m² | 5 |
| € 569.500 | 170 m² | 310 m² | 6 |

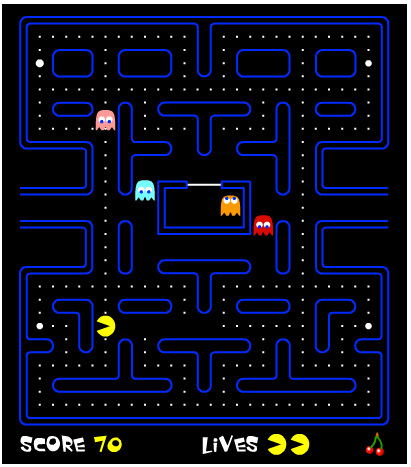
Figuur xx Labeled dataset   
Bron: http://www.funda.nl/koop/huizen/   
  
Bij de training dataset van figuur \*\*\*\* is de gegeven input de huiseigenschappen en de gewenste output de huisprijs. Het systeem wordt met deze dataset getraind. Hierdoor leert het een output te produceren die steeds dichter bij de gewenste output ligt. Als er een verband bestaat tussen de huiseigenschappen en de huisprijs, wat waarschijnlijk het geval is, zal het zelflerende systeem na genoeg trainen in staat zijn zelf bij nieuwe huiseigenschappen een huisprijs te voorspellen. [2]

## Unsupervised Learning

Unsupervised learning kan gebruikt worden bij een ***unlabeled dataset***ofwel, een dataset waarbij de data niet geclassificeerd is en er geen gewenste output bekend is. Als je een dataset hebt van heel veel niet-geordende foto’s is het niet mogelijk om dit te classificeren. Als een deel van de dataset gelabeld wordt, zal met behulp van supervised learning de rest van de dataset geclassificeerd kunnen worden. Dit is echter in veel gevallen niet mogelijk, bijvoorbeeld doordat de dataset enorm groot is of er zodanig veel verschillende groepen bestaan dat het menselijk niet mogelijk is ook maar een deel te labelen. Ook kan het zo zijn dat men niet weet of er een verband aanwezig is.   
Kortom: unsupervised learning wordt gebruikt voor het classificeren van data, zonder dat er groepen vooraf gedefinieerd zijn. Met behulp van deze vorm van training zullen in een grote dataset verbanden kunnen worden ontdekt, die men misschien niet zonder hulp had kunnen achterhalen.[3]

## Reinforcement Learning

Reinforcement learning is een zeer specifieke soort van leren. Er is bij deze vorm van learning geen dataset met input data, maar is er een bepaalde **context**. In deze context bevindt zich een ***agent****.* Een agent is een object dat bepaalde opdrachten kan uitvoeren. De context is een *wereld* waarin deze agent zich bevindt. Door de agent bij bepaalde acties pluspunten of minpunten te geven kun je bepaald gedrag bevorderen.



Figuur Pacman   
Bron: https://www.flickr.com/photos/methodshop/4865516413

In figuur \*\*\*\* is het spel pacman te zien. De agent is hierbij pacman, deze kan namelijk een object dat bepaalde opdrachten kan uitvoeren zoals: loop naar links, loop naar rechts. De context is hierbij het level, ofwel: de positie van de muren (de blauwe obstakels), de posities van de *ghosts (de geleurde vijanden)*, de posities van de pac-dots (de kleine stipjes, ofwel punten) en de posities van de power-pellets (de grotere stipjes). [4] Het eten van de pac-dots is positief, het geraakt worden door de ghosts is negatief. Door reinforcement learing toe te passen op het spel zal de agent steeds beter worden in het spelen van het spel.

# Conclusie

Zelflerende computersystemen zijn algoritmes gebaseerd op machine learning. Een zelflerend systeem verschilt van reguliere algoritmes zoals breadth-first search en depth-first search doordat ze in staat zijn zichzelf te verbeteren.

# Bronnen

[1] Bron: <https://www.cims.nyu.edu/~munoz/files/ml_optimization.pdf>Geraadpleegd op: 21-5-2017  
Laatst gewijzigd op: niet bekend

[2] Bron: <https://www.coursera.org/learn/machine-learning/lecture/1VkCb/supervised-learning>  
Geraadpleegd op: 21-5-2017  
Laatst gewijzigd op: niet bekend

[3] Bron: <https://www.coursera.org/learn/machine-learning/lecture/olRZo/unsupervised-learning>  
Geraadpleegd op: 21-5-2017  
Laatst gewijzigd op: niet bekend

[4] Bron: <http://pacman.wikia.com>  
Geraadpleegd op: 27-5-2017  
Laatst gewijzigd op: niet bekend

## Ordinary Least Squares

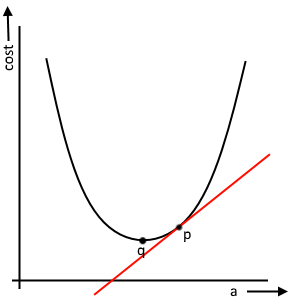
Oridinary Least Squares is een algoritme uit de statistiek dat probeert een linear verband tussen datapunten te vinden. Het probeert een lijn: y = ax + b te vinden die zo goed mogelijk het verband beschrijft tussen de waarden. Om de richtings coëfficiënt (de *a* van de lijn) te berekenen wordt de volgende formule gebruikt:

In deze formule is de gemiddelde van alle x waarden en het gemiddelde van alle y waarden. Om de beginwaarde (b van de lijn) te berekenen gebruiken we de volgende formule:

# Verbeteren

In de vorige deelvraag hebben we verschillende Machine Learning algoritmes behandeld. Hierbij hebben we nog niet besproken hoe een algoritme zichzelf kan verbeteren: Hoe bepalen we bij Linear Regression de waarden voor a en b in de formule y = ax + b? Hoe bepalen we de waarden voor x en b in de vector bij een Support Vector Machine? Hoe bepalen we de wegingen van de synapsen in een Artificial Neural Network? Er zijn verschillende manieren waarop al deze waarden bepaald kunnen worden: Evolutionary Improvement, Newton’s Method en Gradient Descent. Deze drie leerstrategieën zullen we in deze deelvraag behandelen.

# Gradient Descent

De laatste leerstrategie die we behandelen is **Gradient Descent.** Gradient Descent is een algoritme dat functies minimaliseert. Door het aanpassen van bepaalde parameters wordt geprobeerd de waarde van een bepaalde functie zo laag mogelijk te maken. De functie die we bij een zelflerend systeem proberen te minimaliseren is de **cost** **function**, ook wel **loss function** genoemd. Dit is een functie die bepaalt hoe goed het systeem op dat moment werkt. Er wordt bepaald hoeveel de huidige outputs afwijken van de gewenste outputs. Het is hierbij dus nodig dat je de gewenste outputs weet bij gegeven inputs. Er is dus bij gradient descent altijd sprake van supervised learning.   
  
Het algoritme

Figuur 1 - De cost function

In figuur **\*\*\***  is de cost van een bepaalde situatie uitgezet tegen een variabele a. Dit kan bijvoorbeeld de a uit de formule bij Linear Regression zijn. Het is dus te zien dat de cost minimaal is in punt q. We willen dus dat a gelijk wordt aan de waarde van a in punt q. Nu is dit punt in deze grafiek erg makkelijk te vinden, maar zodra er gebruik wordt gemaakt van ingewikkeldere algoritmen, zoals een ANN, wordt dit punt moeilijker te bepalen.  
Op een gegeven moment in het trainingsproces is de a gelijk aan het punt p. Het Gradient Descent algoritme doet dan het volgende:

* De afgeleide op het huidige punt wordt bepaald (de rode lijn in figuur \*\*\*\*).
* De a wordt zodanig aangepast dat het meer in de richting komt van de q. (Dit wordt gedaan door de afgeleide bij de variabele op te tellen)

Wanneer Gradient Descent wordt toegepast zal een bepaalde variabele in een zelflerend systeem zo aangepast worden dat de cost als gevolg van die variabele het laagst wordt.

De wiskunde achter gradient descent  
Het Machine Learning algoritme produceert met een bepaalde input een bepaalde output, dit noemen we de **guess**. Omdat we weten wat de goede ouput is kunnen we de **error** bepalen voor die input. De goede output in de volgende formule is y.

De vorige formule geldt dus voor de individuele datapunten. De totale error, de som van alle individuele error waarden, ook wel cost of loss genoemd kan als volgt beschreven worden:

Zoals bekend uit de wiskunde is het mogelijk om hiervan de laagste waarde te bepalen door de afgeleide op nul te herleiden. Voor elk individueel datapunt is de afgeleide van de error:

Bij het differentiëren wordt gebruik gemaakt van de kettingregel.

Linear Regression met gradient descent  
Om het principe van Gradient Descent beter te begrijpen gaan we nu doormiddel van Gradient Descent Linear Regression uitvoeren. **LEES EERST STUKJE OVER LINEAR REGRESSION**

De guess is hier dus de huidige uitkomst van

De waarde van bi, x­i en yi zijn hier constant. De x­i en yi zijn namelijk bekend uit de training data en bi veranderd wel, maar niet hierbij. De afgeleide van de error functie is dan:

De afgeleide van de cost function is dan:

Met de deze afgeleide is de helling van de cost function te bepalen. Hiermee dus te bepalen welke richting we de variabele a in moeten veranderen. Het aanpassen van de a bij Linear Regression gebeurd dus als volgt:  
a = a + ()  
  
Learning rate  
Een zelflerend systeem bereikt niet in een keer de gewenste output. Er wordt langzaam in de richting van de goede output gewerkt. De formule voor het aanpassen van de *a* waarde uit het vorige kopje is daarom iets anders. Er wordt een learning rate geïntroduceerd:

a = a + ( \* learning\_rate)

Het kiezen van een goede learning rate is heel belangrijk. Een te lage learning rate zorgt ervoor dat het heel lang duurt voordat de goede output bereikt wordt. Een te hoge learning rate zorgt ervoor dat de gewenste output voorbij wordt geschoten. De gewenste output wordt dan nooit bereikt omdat de variabele net te groot of te klein wordt gemaakt. [1,2]

# Bronnen

###### [1] <https://www.coursera.org/learn/machine-learning/lecture/kCvQc/gradient-descent-for-linear-regression> Auteur: Andrew Ng Bezocht op: 17-7-2017 [2] https://spin.atomicobject.com/2014/06/24/gradient-descent-linear-regression/ Auteur: [Matt Nedrich](https://spin.atomicobject.com/author/nedrich/)

Bezocht op: 17-7-2017