# Data Advanced

# Machine Learning

# - IT

***Lector:***
*Heidi Tans*

# 1. Inleiding

Machine Learning is zo'n term die je tegenwoordig overal hoort. Wat is het nu precies? Moet je een wiskundige / statistische achtergrond hebben om aan ML te doen?

Machine Learning (ML) leert machines zelf taken uit te voeren. Zo simpel is het. De complexiteit komt met de details. ML komt in principe neer op het vermogen om zich aan te passen aan nieuwe situaties. Iedere situatie brengt informatie voort en ML wordt gebruikt om hier patronen in te ontdekken en te gebruiken. De machine stelt door middel van algoritmes zelf bepaalde regels op, om bepaalde input te koppelen aan bepaalde output. ML is hierdoor een onmisbaar onderdeel van het onderzoeksgebied van kunstmatige intelligentie, omdat ML ervoor zorgt dat een machine kan evolueren.

In deze introductie tot ML nemen we 4 soorten ML onder de loep.

- Classificatie: Problemen classificeren in voor gedefinieerde categorieën
- Regressie: Verbanden zoeken tussen variabelen
- Clustering: grote datasets clusteren in (betekenisvolle) groepen
- Recommendations: Aanbevelingen doen (aan een gebruiker)

Vooraleer hier aan te beginnen is het belangrijk problemen waar ML een oplossing kan bieden, te herkennen. Eens dit bekeken, zoeken we een oplossing voor sommige problemen op basis van ML technieken.

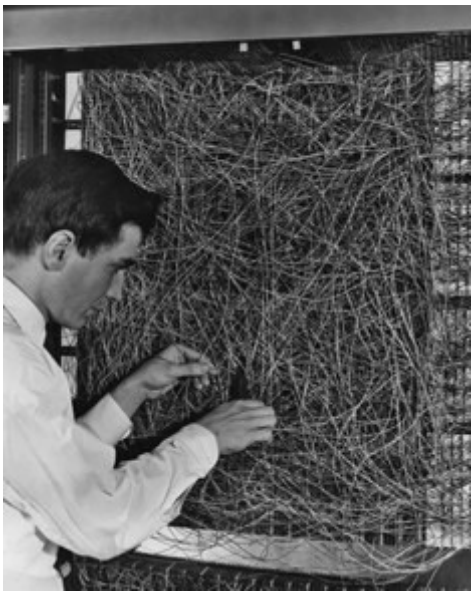
# 2. Historiek

Arthur Samuel, een werknemer van IBM, is degene die de term *machine learning* introduceerde (rond 1952). Hij werd als werknemer van zijn bedrijf gezien als een autoriteit en pionier op het gebied van gaming en kunstmatige intelligentie. Hij gebruikte hier een schaakspel voor, dat steeds “slimmer” werd al naar gelang het meer speelde. Het spel onthield winnende zetten en gebruikte deze in zijn eigen partijen.

Dit was het begin van de uitwerking van het concept van *machine learning* in het algemeen, en van de schaakcomputer in het bijzonder: het wordt dan ook regelmatig aangemerkt als zijnde de grootvader van Deep Blue, de IBM-computer die in 1997 schaakkampioen Garri Kasparov versloeg.

Na Samuel volgden nog veel innovaties die allen verder gingen op het vinden van algoritmes die zichzelf leren aan de hand van de verzamelde data, en die op basis hiervan in staat zijn tot zelfstandige besluitvorming.

Een hoogtepunt hiervan vond plaats in 1958, het jaar waarin Frank Rosenblatt de Perceptron ontwierp. Dit was het eerste kunstmatige neurale netwerk – een kopie van het menselijke neurale netwerk – wat de basis vormde voor meer menselijke intelligentie in computers. Aan de hand van stimuli (input) werd een analyse uitgevoerd, en vervolgens omgezet in een reactie (output).



Ook noemenswaardig is het project van studenten van Stanford University in 1979, die tot de ontwikkeling van de zogenaamde ‘Stanford Cart’ leidde: een bewegende robot die in staat was om zelfstandig door een kamer te bewegen en onderweg obstakels te ontwijken. Een verre voorvader van zelfrijdende auto’s en robot-stofzuigers, dus.

# 3. Machine Learning problemen herkennen

Door een aantal voorbeelden te overlopen, krijg je geleidelijk aan intuïtief voeling wat ML is en waar het kan gebruikt worden. Bij problemen die zich in de toekomst voordoen, denk je na dit deel misschien ook even aan ML om jou te helpen met het vinden van een oplossing. ML blijkt de drijvende kracht te zijn achter heel wat applicaties uit het dagelijkse leven die wij eigenlijk “vanzelfsprekend” noemen.

**Een typisch, klassiek ML voorbeeld**

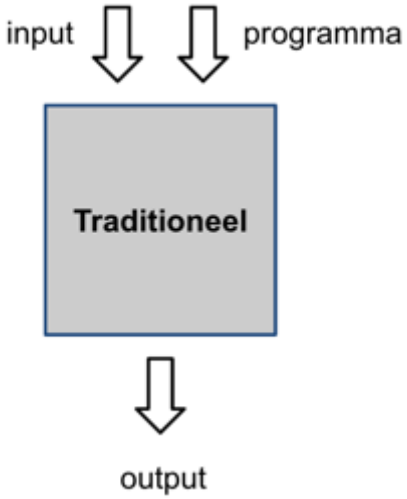
Je bent een “alien” en kijkt naar de bevolking ergens op de wereld en je weet dat er 2 soorten mensen op de wereld zijn: zeg man en vrouw. Maar je kent de karakteristieken om te differentiëren tussen man en vrouw niet.

Maar je kunt wel zien dat mensen die groter zijn dan bvb 1.8 tot de groep van de mannen behoren. De andere groep zijn dan de vrouwen.

***Je zou dit probleem kunnen oplossen op volgende "klassieke" manier:***

```
if lengte > 1.8:
    geslacht = "man"
else:
    geslacht = "vrouw"
```

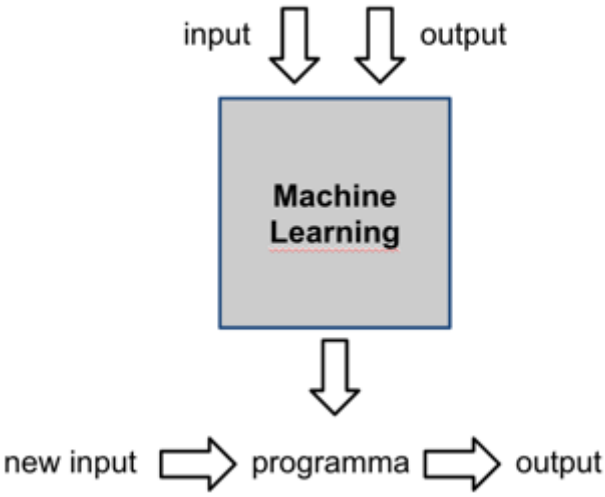
Je steekt data én een programma in een computer en je verkrijgt output; dit is een benadering van het probleem op basis van regels; op basis van programmatielogica.



***ML benadert dit probleem op een andere manier:***

Die alien kidnapt jou en jij kan per persoon aanduiden of het een man of een vrouw is. Nadat die alien geobserveerd heeft hoe jij een 100- of 1000-tal mensen classificeert als man of vrouw (= dit noemen we de training data set); leert hij "intuïtief" hoe te differentiëren tussen geslachten.

Deze ML benadering is vrij gelijklopend met "hoe" mensen leren; hoe het menselijk brein werkt. Hoe leerde je vroeger als kind? Je herkende patronen. Je werd voor langere tijd blootgesteld aan bepaalde fenomenen en je leerde door ervaringen.




**"Definitie ML": ML is het proces waarbij een computerprogramma of systeem in staat is te leren hoe een taak uit te voeren en dit door ervaring. Ervaring voor een computer is data!**

**Voorbeeld**

- Netflix kan hun service personaliseren naar elke individuele gebruiker toe want zij lijken te weten waar wij in geïnteresseerd zijn.
- Google Maps: we kunnen er niet meer zonder; deze service maakt je dagelijks leven zoveel makkelijker.
- Spelling checker
- e-mail programma: bepaalde e-mails worden als “spam” geclassificeerd. Hoe komt dit?
- De zelfrijdende auto die geheel autonoom van punt A naar B kan, onderweg rekening houdend met alle variabelen – zoals over verkeer, omstandigheden, obstakels, verkeersregels, etc.

- De slimme thermostaat: Deze slimme thermostaat voor in huis leert van de gebruiker die de thermostaat bedient. Wanneer deze de thermostaat bijvoorbeeld iedere maandag, woensdag en vrijdag rond een uur of zes ‘s avonds op 21 graden zet, dan leert de thermostaat zich dit schema aan. Hierdoor zorgt hij ervoor dat het iedere maandag, woensdag en vrijdag rond zes uur ‘s avonds 21 graden is; zonder tussenkomst van de gebruiker.


- Bio-informatica waarmee onder meer eiwitfuncties voorspeld kunnen worden
- Natuurlijke taalverwerking zodat bvb de functie en betekenis van woorden in een zin bepaald kunnen worden en effectievere vertaal- en schrijfprogramma’s ontworpen kunnen worden.
- Gezichts- en stemherkenning die aan de hand van visuele of audio-input bepaalde kenmerken kan opslaan en dit matchen aan bijpassende output. Een gezichtherkenningssysteem is hier een uitstekend voorbeeld van, waarbij input van talloze bewakingscamera’s uitgelezen wordt om gelijkende matches te vinden met een vooraf opgegeven profiel.
- Data ivm “user clicks and views”. Deze worden gebruikt om aanbevelingen te doen over dingen die een gebruiker zou kunnen willen of nodig heeft.

Deze voorbeelden lijken een soort intelligentie van “hun eigen” hebben. Maar het is ML dat hen brengt tot deze zogeheten intelligentie.

Voor ML heb je dus veel data nodig. Gelukkig wordt er tegenwoordig constant data verzameld vandaar dat ML een heel breed toepassingsgebied heeft.

Dus… de volgende keer als zich een probleem voordoet waarbij je de computer een taak dient te leren en je hebt veel data, stel jezelf ook eens de vraag of ML dit probleem misschien kan oplossen ipv rechttoe rechtaan programmeertechnieken toe te passen.

# 4. Wanneer Machine Learning gebruiken?

Neem het voorbeeld van google Maps. Dagelijks gebruiken veel mensen google maps om van één punt tot een ander te geraken. Hoe lang duurt dit? Wat is de snelste weg? Welke obstakels bevinden zich op dat moment op de route?

Google maps geeft je niet alleen de weg maar ook hoe lang het duurt op een gegeven moment van de dag (dus rekening houdend met eventuele files..)

Stel dat je als opdracht google maps moet "nabouwen". Dan zijn er 2 manieren om dit probleem te benaderen.

***Benadering op basis van regels (algoritme):*** er zal een grote set regels opgesteld moeten worden om dit probleem te programmeren; 1 simpele regel zou kunnen zijn:

```
if dag == "maandag" & tijdstip == 1 & afstand == 4:
    duur = 12
```

Dit is slechts 1 regel. Je moet dit doen voor elke dag, voor elk tijdstip, elke afstand,… Je hebt nood aan 1000-den regels. Deze regels zijn opgesteld door mensen na heel veel research. Dit is zowel tijd- als kost rovend. Een bijkomend nadeel hierbij is dat deze regels **statisch** zijn: ze wijzigen zelden en als ze al wijzigen, is dit traag in tijd. Deze wijzigingen moeten weeral gemaakt worden door analisten, gebaseerd op nieuw onderzoek (tijd, geld, …)…

Een statische aanpak voor dit soort probleem is eigenlijk niet aangewezen aangezien verkeerssituaties dynamisch zijn: het weer, is het een vakantiedag of niet?, … zijn ook factoren die dit beïnvloeden. Dus simpele regels op basis van basisfactoren voldoen waarschijnlijk niet om dit probleem op te lossen.

***Benadering mbv ML***: Deze benadering bestaat uit 3 stappen:

1. Verzamel een grote set data ivm verkeersgegevens (10 of zelfs 100 verschillende variabelen zouden het verkeer kunnen beïnvloeden en je moet deze allemaal verzamelen) en blijf deze ook verzamelen op een continue manier.
2. Gebruik een algoritme dat "zelfstandig" het verband vindt tussen deze variabelen en de duur van de gevraagde trip
3. Update dit verband voortdurend mbv nieuwe data

Wat is dus het verschil?

**Rule based**: je past een aantal statische regels toe op de huidige context (tijdstip, dag, …) waaruit de tijdsduur berekend wordt.

**ML**: Ook in de ML benadering pas je een aantal regels toe maar het verschil is dat deze regels (automatisch) geüpdatet worden op basis van nieuwe data.

***Wanneer gebruik je ML dan? Enkele vuistregels:***

- Wanner het moeilijk is om het probleem te gieten in regels owv het grote aantal variabelen die de outcome beïnvloeden en de moeilijkheid om de relatie tussen de variabelen te snappen
- Een voorwaarde om ML benadering toe te passen is dat je beschikt over een grote set historische data
- De patronen of relaties tussen de data zijn dynamisch (blijvend veranderend onder andere omstandigheden)

Let op: ML is vaak aantrekkelijk om toe te passen maar het mag zeker niet zomaar vanaf nu de default keuze worden.

# 5. Het ML proces

Op basis van het voorgaande, kunnen we beslissen of we bij een specifiek probleem voor de ML aanpak gaan of niet. Wanneer we ons probleem met ML gaan oplossen volgen we een gestructureerd plan om tot de oplossing te komen

5.1. Bepaal welk type ML probleem we hebben

5.2. Gebruik de data die je hebt

5.3. Pas een standaard algoritme toe op jouw data

Wanneer je deze 3 stappen volgt, kan je elk ML probleem oplossen.

## 5.1 Type ML – problemen

Het type ML - probleem is onder te verdelen in 2 grote categorieën: Supervised Machine Learning en Unsupervised Machine Learning.

Bij Supervised Machine Learning hebben we training data voorhanden waar zowel **input als output** aanwezig is. Bij Unsupervised Machine Learning beschikt het algoritme enkel van training data die voorzien is van input (geen output).

Voorbeelden van Supervised Learning:

5.4. Classificatie

5.5. Regressie

Voorbeelden van Unsupervised Learning:

5.6. Clustering

5.7. Recommendations

## 5.2 Data

Wanneer je voor ML gekozen hebt, heb je een grote hoeveelheid historische data nodig. Deze data kan beschikbaar zijn in ongestructureerde tekst, video's, foto's, … Maar om deze data te gebruiken in een ML context is het nodig om deze data om te zetten naar betekenisvolle numerieke data. Bvb een foto: je kan de hoogte en breedte van de foto gebruiken en een numerieke waarde die de kleurintensiteit van elke pixel in de foto weergeeft. Het bepalen van betekenisvolle kenmerken (en die omzetten in numerieke waarden) is cruciaal voor het oplossen van het probleem.

## 5.3 ML - algoritme

Tot slot jaag je de verzamelde data doorheen een algoritme. Je gebruikt de verzamelde data (historische data) om patronen te ontdekken; patronen in de vorm van regels.

Deze regels tonen de verbanden tussen variabelen aan. Alle regels samen die het algoritme gevonden heeft, wordt een model genoemd. In het voorbeeld om e-mail te classificeren als spam of niet, representeert het model de relatie die er is tussen bepaalde tekst in een e-mail en het al dan niet spam zijn. Een model kan een wiskundige vergelijking zijn of een verzameling van regels. Welke algoritme er gekozen wordt, hangt er af van het type probleem:

- Classificatie ➔ Naive Bayes algoritme, Support Vector machine algoritme, …
- Clustering ➔ K – means clustering algoritme, K – median clustering algoritme, Density-Based Spatial clustering algoritme, Hierarchical clustering, …

In tegenstelling tot wat doorgaans gedacht wordt, is het bepalen van het algoritme niet het meest bepalende deel in het oplossen van ML: je kan via "plug and play" meerdere bestaande algoritmes met een minimum aan inspanning uitvoeren op jouw probleem aangezien heel wat programmeertalen "build in packages" hebben waarin deze algoritmes voorzien zijn. Dus… het kiezen van het algoritme kan voortkomen uit "trial and error". De setup van het probleem en het juist presenteren van de data, vergt vaak het meeste energie.

# 6. Types van ML problemen

Een cruciale stap is het identificeren welk type ML probleem we hebben. Dit hangt af van hoe je het probleem definieert. Deze beslissing nemen houdt in dat je een aantal doordachte keuzes moet maken. Je kan een probleem hebben en benaderen als een classificatie maar ook als een regressie probleem. We gaan nu door elk type van ML probleem (classificatie – regressie – clustering en recommendations) en we kijken welke types van probleemsituaties onder elk van deze categorieën gedefinieerd kunnen worden.

## 6.1 Classificatie

Alle classificatie problemen hebben een aantal gemeenschappelijke karakteristieken:

- Doel: Een object (= problem instance) classificeren
- De categorieën waaraan we een object toekennen, zijn op voorhand gekend. 2 categorieën noemt men binary classification maar er kunnen ook meerdere categorieën zijn.
- Classifier = algoritme dat de indeling uitvoert. Deze classifier gebruikt data waarvan de categorie reeds gekend is. Voorbeelden van classifiers = naieve bayes algoritme, support vector machine algoritme
- Training data is de data die al geclassificeerd is. Van deze training data leert onze classifier hoe nieuwe data te classificeren

Voorbeelden:

- Reclame detectie
    - Doel: Is deze advertentie (= problem instance) reclame of niet?
    - Categorieën: reclame of geen reclame (dus binary)
    - Classifier: Support vector machine algoritme
    - Training data = advertenties die al geclassificeerd zijn als reclame of geen reclame

- Sentiment analysis
    - Doel: Is de tweet (= problem instance) positief of negatief?
    - Categorieën: positief of negatief
    - Classifier: Naive Bayes algoritme
    - Training data = tweets die al geclassificeerd zijn als positief of negatief

- Trading strategy
    - Doel: Zal een dag (= problem instance) op de beurs positief of negatief zijn?
    - Categorieën: positieve dag of negatieve dag
    - Training data = dagen die reeds als positief of negatief geclassificeerd zijn

- Spam detectie
    - Doel: Is deze e-mail (= problem instance) spam of niet?
    - Categorieën: spam of geen spam
    - Training data = e-mails die al geclassificeerd zijn als spam of geen spam

## 6.2 Regressie

In regressie voorspellen / berekenen we een continue variabele (= output). Dit in tegenstelling tot classificatie waar de output categorical is. Verder weten we bij regressie dat deze output afhangt van/beïnvloed wordt door sommige andere variabelen (input).

In regressie geven we antwoorden op vragen als:
Hoe lang duurt het om van punt A naar punt B te gaan (google maps)?

- De afhankelijke (continue) variabele = tijdsduur
- Onafhankelijke variabelen: tijdstip op de dag, afstand, …

Hoeveel producten zullen we verkopen in deze week?

- De afhankelijke (continue) variabele = aantal producten dat verkocht zal worden
- Onafhankelijke variabelen: verkoop vorige week, prijs, vakantie, …

We kunnen dit zien als: we hebben een aantal variabelen die de output beïnvloedt, hier passen we één of andere functie op toe en als uitkomst krijgen we de voorspelde waarde van onze afhankelijke variabele.
Het regressie algoritme moet het verband zoeken tussen die variabelen:
Is dit verband lineair, kwadratisch, exponentieel, … ?

Net zoals bij classificatie heeft regressie ook nood aan training data. Deze training data is een grote dataset waarbij zowel input als de output (= de waarde die voorspeld moet worden) aanwezig is.

## 6.3 Clustering

Een voorbeeld van clustering: stel je hebt een grote groep mensen die actief zijn op social media. Op basis van gemeenschappelijke kenmerken willen we deze gebruikers indelen in groepen zodanig dat in één groep de gebruikers "vergelijkbaar" zijn met elkaar.

Belangrijk hierbij is dat de groepen waarin deze gebruikers ingedeeld gaan worden, op voorhand NIET gekend zijn! Dit is anders dan in classificatie: hier begin je met een bepaalde set groepen waarin je de objecten plaatst. We gebruiken het clustering algoritme om te ontdekken/inzicht te krijgen in welke groepen er kunnen bestaan.

In het voorbeeld van de gebruikers van social media: het algoritme deelt de gebruikers op in clusters. Door deze clusters te bestuderen, kan je later misschien afleiden dat elke cluster één of andere betekenisvolle groep vertegenwoordigt: demografisch, voorkeur van de gebruikers, leeftijd, …

Je gebruikt clustering dus om te ontdekken welke betekenisvolle groepen er bestaan binnen bvb alle gebruikers van facebook. Wanneer dan een nieuwe gebruiker zich aanmeldt op FB, dan kan je classificatie gebruiken om deze user toe te kennen tot één of andere groep die ontstaan is door clustering.

## 6.4 Recommendations

Bij dit soort van ML probleem gaan we aanbevelingen (voorspellingen van bvb gedrag) doen. Op basis van eerder gedrag (opzoeken op internet, luisteren naar bepaalde liedjes, aankopen in een grootwarenhuis, …) worden items waar de gebruiker mogelijks nog in geïnteresseerd is, voorspeld; zelfs nog voor hij/zij het zelf al beseft.

Voorbeeld:

- Een lezer van boeken ➔ boeken aanraden
- Een koper van een telefoon ➔ wat kan nog handig zijn om erbij te kopen (hoesje…)

De key hier is: neem gedrag uit het verleden en bepaal van daaruit wat de user nog nodig kan hebben of graag heeft. Deze techniek noemt met Collaborative filtering.

**Definitie: Collaborative filtering** is een techniek, waarbij men gebruik maakt van historische gegevens, die verzameld zijn van andere gelijke gebruikers of bezoekers, met als doel het doen van aanbevelingen voor een nieuwe klant op basis van voorkeuren van andere klanten met een vergelijkbaar klantprofiel.

Op basis van bovenstaande kan je een nieuw probleem indelen in één van bovenstaande 4 ML problemen. De keuze die je hier maakt, bepaalt volledig wat er zal gebeuren in de volgende stappen.

# 7. ML problemen uitgewerkt

## 7.1 Supervised ML: Classification

We bekijken even de verschillende stappen die bij elk classificatie – probleem opduiken.

1. Definieer het classificatie - probleem (=classification problem statement). Een “problem instance” is een entiteit die we willen classificeren. We gooien dit object door de “classifier” en die geeft een label aan het object. Beschouw de classifier voorlopig als een black box.

Het doel van ML = om deze classifier te bouwen! Wat binnen in die black box gebeurt, kunnen we ons voorstellen als wiskundige of statistische algoritmes (regels of vergelijkingen). Dit noemen we een model.

2. Features: We moeten data verzamelen die van numerieke aard is en representatief voor het probleem dat je wil oplossen. Zelfs als je tekst of foto’s gebruikt als data voor je ML probleem, moet je deze gegevens omzetten naar een numerieke grootheid.
3. Training: train een model gebruik makend van training data. Neem een grote hoeveelheid historische data die reeds correct gelabeld is. Het classificatie algoritme leert van de training data. Het tracht regels en patronen af te leiden van de training data. Dit noemen we het model. De training data kunnen we zien als een tupel van features én label.
4. Testing: test het model gebruik makend van test data. In deze fase classificeren we nieuwe data die we nog niet gezien hebben in één van de categorieën. De efficiëntie van onze classifier hangt af van de juistheid van het classificeren van objecten die niet in de training data zitten.

Stap 1 en stap 2 zijn cruciaal in het juist opzetten van het ML - probleem en is erg tijdrovend! Eens dat je jouw probleem en features correct bepaald hebt, dan kan je voorgeprogrammeerde algoritmes loslaten op jouw training data en later op nieuwe objecten. Het grote voordeel is dat deze voorgeprogrammeerde algoritmes beschikbaar zijn in libraries van bvb python, R, Sparc.

Vb van bestaande algoritmen: Naive Bayes algoritme, suport vector machine, decision trees, K-nearest neighbour, random forests, logistic regression, …

**Classificatie: Uitgewerkt voorbeeld sentiment analyse (tekst classificeren op basis van gevoel) mbv Naive Bayes algoritme**

Hoe iemand zich voelt (positief of negatief) kan gemeten worden met een techniek die we Sentiment analysis noemen. Deze techniek bestaat al lang maar is de afgelopen jaren erg populair geworden omdat iedereen te pas en te onpas op social media zijn gevoel, bedenkingen, … post. Dus er is onnoemelijk veel data voorhanden hier. De data bestaat vooral uit tweets, meldingen, emoji's, … Deze data is vrij ongestructureerd maar wel publiekelijk toegankelijk voor iedereen.

Herinner:

- Sentiment analysis
    - Doel: Is de tweet (= problem instance) positief of negatief?
    - Categorieën: positief of negatief
    - Classifier: Naive Bayes algoritme
    - Training data = tweets die al geclassificeerd zijn als positief of negatief

1. Definitie van het classificatie probleem: we hebben een commentaar en we moeten deze classificeren als zijnde ofwel positief ofwel negatief; dit noemen we ook wel het identificeren van de polariteit van een commentaar.
2. Features om de data te representeren zijn hier niet zo evident: we hebben ongestructureerde data en voor een classificatieprobleem hebben we nood aan numerieke maatstaven. Een mogelijkheid om teksten om te zetten naar numerieke data is dmv **"term frequency representation"**: Dit is een erg populaire manier om tekst om te zetten naar numerieke waarden om zo te kunnen gebruiken in ML algoritmes.
   We maken een lijst van ALLE woorden die in ALLE commentaren of teksten of … kunnen voorkomen.

   $$(woord_1 , woord_2, \dots , \dots , \dots , \dots , \dots , \dots , \dots , \dots , \dots , \dots , woord_N )$$

   *(hallo, dit, zijn, alle, woorden, die, kunnen, voorkomen, in, een, tekst, en, ook, is)*

   Hiermee kunnen we om het even welke tekst voorstellen door een lijst van getallen die de frequenties voorstellen van elk woord dat in de tekst voorkomt. Bvb: volgende zin:

   *(hallo, dit, is, een, tekst, en, een, tekst)*

   *(hallo, dit, zijn, alle, woorden, die, kunnen, voorkomen, in, een, tekst, en, ook, is)*

   De testzin wordt dan voorgesteld door volgend N - tupel:

   *(1, 1, 0, 0, 0, 0, 0, 0, 0, 2, 2, 0, 0, 1)*

   Hiermee zien we dat "hallo" 1 keer in de zin voorkomt en "tekst" 2 keer in de zin voorkomt. Wanneer een woord niet voorkomt in de zin is de frequentie uiteraard 0.
   Merk op: Aangezien elk document anders van lengte is, is het mogelijk dat éénzelfde term in lange documenten veel vaker voorkomt dan kortere. Zo wordt de term frequentie vaak gedeeld door de documentlengte (ook wel het totale aantal termen in het document) als een manier om te normaliseren. Voor voorgaand voorbeeld zou dit worden:

   *(1/8, 1/8, 0, 0, 0, 0, 0, 0, 0, 2/8, 2/8, 0, 0, 1/8)*

3. De training fase: hier maken we gebruik van commentaren die al gelabeld zijn als zijnde positief of negatief.
4. De test fase

**Het naive bayes algoritme werkt als volgt:**

De training data set bestaat uit een groot aantal teksten (commentaren of tweets of …) die reeds “positief” of “negatief” gelabeld zijn.

| | |
|---|---|
| **tekst 1** | *Positief* |
| **tekst 2** | *Positief* |
| **tekst 3** | *Negatief* |
| **tekst 4** | *Positief* |
| **tekst 5** | *Negatief* |
| **tekst 6** | *Positief* |
| … | |
| **tekst 1000** | *Positief* |

Hiermee berekent men de kans op een positieve tekst en wel als volgt:

$$P_O = \frac{\#\ positieve\ comments\ in\ de\ training\ data}{\#\ comments\ in\ de\ training\ data}$$

en owv de complementregel weten we dan dat de kans op een negatieve tekst gelijk is aan $1 - P_O$ Neem voor de verdere uitrekening $P_O = 55\%$ en $1 - P_O = 45\%$

Met behulp van “term frequency representation” zetten we elke tekst om naar een tupel van cijfers. Elk cijfer stelt het aantal keer voor dat het specifieke woord in het geheel van de training data voorkomt. Dit noemt men de woordfrequentie.

| | |
|---|---|
| **(1,0,1,1,…,0)** | *Positief* |
| **(1,1,2,1,…,1)** | *Positief* |
| **(2,0,1,0,…,0)** | *Negatief* |
| **(0,0,1,1,…,2)** | *Positief* |
| **(1,1,1,1,…,0)** | *Negatief* |
| **(1,2,0,0,…,0)** | *Positief* |
| **…** | |
| **(1,0,2,2,…,3)** | *Positief* |

Teksten zijn samengesteld uit zinnen, die op hun beurt uit woorden bestaan. Van deze individuele woorden wordt vervolgens de positiviteitskans berekend.

We berekenen bvb de positiviteitskans van het woord “blij”:

$$P_{blij} = \frac{Som\ van\ alle\ woordfrequenties\ van\ blij\ in\ positieve\ teksten}{Som\ van\ alle\ woordfrequenties\ van\ blij\ in\ de\ hele\ training\ data}$$

Dit is dus een maatstaf die ons aangeeft van al de keren dat “blij” voorkomt in een tekst, hoe vaak komt het voor in een positieve tekst.

Analoog kunnen we nu de negativiteitskans van hetzelfde woord berekenen als zijnde: $1 - P_{blij}$ .

Dit doen we nu voor elk woord dat voorkomt in de training dataset; wat ons bvb volgende tabel oplevert:

| | ***Positiviteitskans*** |
|---|---|
| **blij** | 0.92 |
| **geweldig** | 0.95 |
| **pxl** | 0.81 |
| **data** | 0.99 |
| **saai** | 0.1 |
| **moeilijk** | 0.33 |
| **…** | |

Ieder woord krijgt een positiviteitskans en dus ook een negativiteitskans (1 – positiviteitskans).

Hoe passen we dit nu toe op de test data om nieuwe teksten te classificeren als positief of negatief?

We nemen een nieuwe tekst bestaande uit een aantal woorden en het algoritme gaat een positiviteits- en negativiteitsscore voor deze tekst berekenen op basis van de waarden uit de training data.

We berekenen dit voor bvb de zin “data is geweldig”

$$PosScore = P_{data} * P_{geweldig} * P_O = 0.99 * 0.95 * 0.55 = 0.51$$

$$NegScore = (1 - P_{data}) * \left(1 - P_{geweldig}\right) * (1 - P_O) = 0.01 * 0.05 * 0.45 = 0.000225$$

We vergelijken deze 2 scores en geven het label met de hoogste score aan de nieuwe zin.

Merk hierbij op dat niet betekenisvolle woorden (zoals is, het, en, een, …) genegeerd worden.

**Waarom wordt dit NAIVE Bayes genoemd?**

Wanneer we de algemene positiviteitsscore berekenen, gebruiken we **onafhankelijk** de positiviteitsscores van de verschillende woorden in het comment. We houden geen rekening met het feit hoe de positiviteitsscore beïnvloed wordt wanneer er paren van woorden optreden. Bvb als er een bepaalde zin in een commentaar optreedt ➔ gaat die zin meer bijdragen tot de positiviteitsscore (is het gebruik van voorwaardelijke kansen aangewezen)? Het naive bayes algoritme veronderstelt dat elke variabele **onafhankelijk** bijdraagt tot de positiviteitsscore. Er wordt geen rekening gehouden met mogelijke interacties tussen woorden.

Dit is best een grote veronderstelling waarbij je zou kunnen denken dat het Naive Bayes algoritme niet erg krachtig is. Dit is niet waar: Naive Bayes is uitermate aangewezen wanneer je

- weinig training data voorhanden hebt
- wanneer je zelf weinig verdere info hebt of kennis in het domein

**Bekijk PLURALSIGHT: How to think about Machine Learning Algorithms:**

**4. Solving Classification Problems – Implementing Naive Bayes (8 minuten)**

**Classificatie: Uitgewerkt voorbeeld Reclame blokker voor websites mbv Support Vector Machines Algoritme**

Herinner:

- Reclame detectie
  - Doel: Is deze advertentie / beeld (= problem instance) reclame of niet
  - Categorieën: reclame of geen reclame
  - Classifier: Support vector machine algoritme
  - Training data = advertenties die al geclassificeerd zijn als reclame of niet

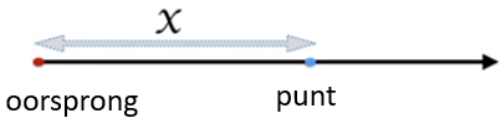
1. Definitie van het classificatie probleem: We willen bepalen of een beeld een reclame advertentie is of niet. Ook dit is een klassiek ML classificatie probleem. Het problem instance = een beeld waarvan we gaan bepalen of het reclame is of niet.

2. Features = beelden die numeriek voorgesteld kunnen worden. We hebben afbeeldingen en we gaan die voorstellen door een tupel van numerieke waarden. We kunnen bvb de hoogte, breedte, url, tekst op pagina, tekst bij de afbeelding gebruiken.

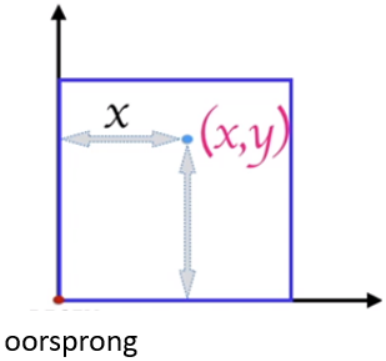
3. De training data bestaat uit een grote data set van afbeeldingen die reeds het label wel of geen advertentie ontvangen hebben.
   Aan al deze afbeeldingen is een tupel van bvb N getallen geassocieerd (zie punt 2).
   Deze punten kunnen voorgesteld worden in een N – dimensionale figuur.

   *Wiskundig intermezzo N – dimensionale figuur/ ruimte:*

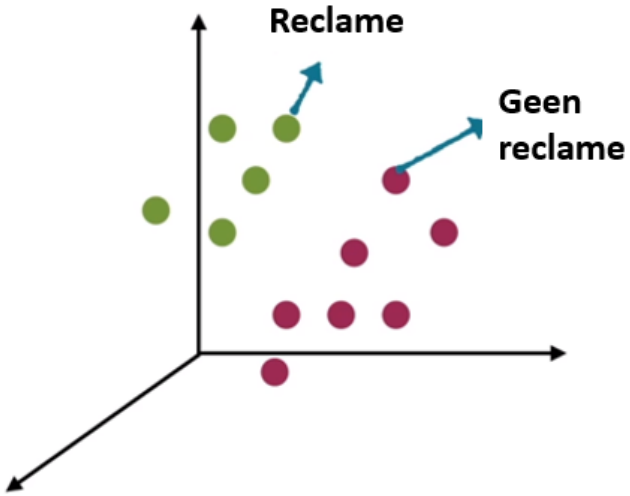
   Een één - dimensionale figuur is een lijn en elk punt op die lijn kan voorgesteld worden door 1 waarde nl de horizontale afstand tot de oorsprong:




   Een twee – dimensionale figuur is bvb een vierkant en elk punt op het vierkant kan voorgesteld worden door een koppel (x , y) met x de horizontale afstand tot de oorsprong en y de verticale afstand tot de oorsprong.

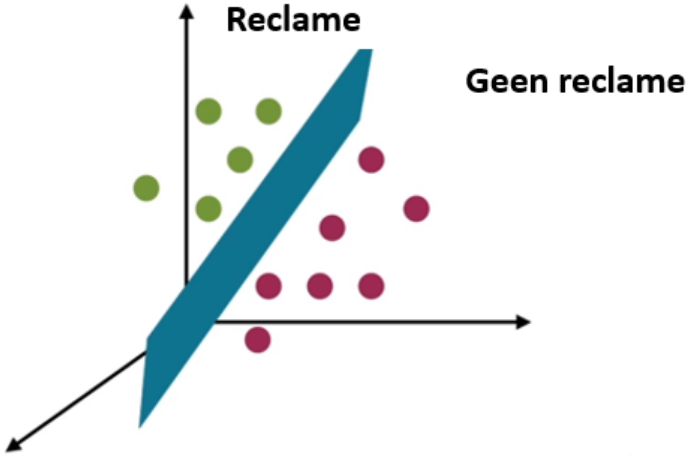


Bij veel ML – problemen wordt er gebruik gemaakt van N – dimensionale figuren. In de training fase zetten we alle figuren van onze training data set uit in een N – dimensionaal assenstelsel en we markeren de reclame figuren bvb als groen en de niet – reclame figuren als rood.

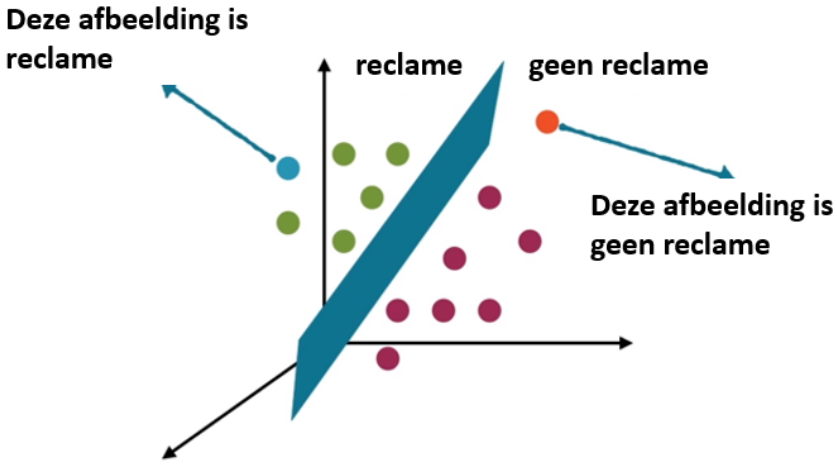



**Wat doet het Support Vector Machine algoritme?**

Het SVM gaat op basis van de punten van de training data een grens – vlak zoeken tussen de 2 groepen.



4. Eens deze grens gevonden, kunnen we verder gaan naar de test – fase waar we nieuwe afbeeldingen willen gaan classificeren als reclame of geen reclame. We zetten de nieuwe figuur uit op het 'assenstelsel' en kijken gewoon aan welke kant van de scheiding dit nieuwe punt valt.




SVM kunnen we **enkel** gebruiken voor **binaire** classificatie (dus in 2 categorieën), dit in tegenstelling tot Naive Bayes waar we kunnen indelen in om het even welk aantal categorieën.

**Bekijk PLURALSIGHT: How to think about Machine Learning Algorithms:**

**4. Solving Classification Problems – Implementing Support Vector Machine**

**(9 minuten)**

**Verdere toepassingen Classificatie: het gedrag van een klant begrijpen om…**

- … te voorspellen of de klant van plan is om van bepaalde services geen gebruik meer te maken. Voor hij/zij de knoop definitief doorhakt is het essentieel om met gerichte acties de klant op andere gedachten te brengen.
- … mogelijke fraude te detecteren: het gedrag van een klant in de gaten houden om te voorspellen of de klant kanshebber is om fraude te plegen en indien ja ➔ weigeren tot jouw services
- … te bepalen of een klant kredietwaardig is of niet: Indien niet kan de financiële instelling beslissen geen lening toe te staan of enkel onder bepaalde garanties, zekerheden, …

**Verdere toepassing Classificatie: het bepalen van het geslacht van een online gebruiker**

Wanneer je online gegevens moet invullen, vul je meestal enkel het echt noodzakelijke in: naam, voornaam, e-mail maar geslacht bvb is niet altijd verplicht en velen laten het dan ook open. Voor vele toepassingen is het wel handig om het geslacht te weten (bvb om gerichte reclame sturen). Het is niet opportuun om ook dit veld verplicht te maken want zoveel te meer iemand MOET invullen, zoveel te minder geneigd hij/zij is om met die site door te gaan. Hoe kan men dan toch het geslacht te weten komen (van mensen die dit niet ingevuld hebben)? Kan het systeem op basis van bvb de voornaam een goede voorspelling doen of het om een man of vrouw gaat? Dit probleem vertaalt zich in een ML probleem van classificatie. We hebben ook hier typisch de vier stappen in classificatie:

1. Definitie van het classificatie probleem: we willen een persoon indelen in man – vrouw (= label) op basis van voornaam

2. Features: we hebben de voornaam (= feature), die willen we door onze black box (classifier) gooien en hieruit zouden we willen afleiden of het een man of een vrouw is. Belangrijk bij classificatie is (zoals geweten) dat we numerieke features hebben. We moeten op de één of andere manier aan onze voornaam een numerieke waarde toekennen. Hiervoor kunnen we karakteristieken van namen gebruiken die meestal een verschil maken tussen mannelijke voornamen en vrouwelijke voornamen bvb vaak eindigen vrouwelijke namen op een klinker. We maken een binaire variabele (1 / 0) met de waarde 1 als eindigend op klinker en 0 als eindigend op medeklinker. Een andere feature zou kunnen zijn: het aantal karakters. Er kunnen sommige pre- of suffixen zijn die specifiek zijn voor een bepaald geslacht. Zo zie je dat er heel wat denkwerk te pas komt aan het bepalen van deze features; welke hebben invloed op de outcome en hoe? Vaak werk je hier via trial and error om te zien hoe zo'n feature kan leiden tot de juiste outcome.

3. Training data set: de mensen die het geslacht wel ingevuld hebben.

4. Pas het algoritme toe op nieuwe data.

## 7.2 Supervised ML: Regressie

Regressie is interessant wanneer je verbanden tussen variabelen wil begrijpen. Hoe kunnen we een regressie probleem definiëren? Wat zijn afhankelijke en onafhankelijke variabelen? Wat is het verschil tussen classificatie en regressie?

**Hoe kan je regressie problemen herkennen?**

Enkele voorbeelden:

- Wat zal het rendement zijn van een aandeel op een bepaalde datum?
- Als de wachttijd vergroot, welk effect heeft dit op klanttevredenheid?
- Wat is de verwachte verkoop van een bepaald product in een bepaalde week?

Er zijn 2 verschillende soorten regressie problemen:

1. We voorspellen een continue variabele (bvb we voorspellen het rendement van een aandeel op een bepaalde datum of de verwachte verkoop)
2. We bepalen het verband tussen 2 variabelen (bvb wat is het effect van een wachttijd op klanttevredenheid)

We focussen op soort 1: het rendement van een aandeel op een bepaalde dag kan afhangen van meerdere variabelen (= onafhankelijke variabelen); bvb dag van de week, dag van de maand, het rendement van de afgelopen dagen, …).

Stel: we hebben gegevens over deze onafhankelijke variabelen, kunnen we een wiskundige functie vinden die het rendement van het aandeel zal berekenen / voorspellen? Dit is wat regressie doet: regressie zoekt de functie die het verband tussen de onafhankelijke en afhankelijke variabelen zo goed mogelijk benadert. Regressie start met het ”veronderstellen” van een bepaalde functie die het verband weergeeft: bvb lineair verband, polynomiaal verband, niet-lineair verband. Lineaire regressie is veruit de meest gebruikte en gekende regressie techniek.

**Vb voorspellen van "vraag" waaruit "aanbod" bepaald wordt:**

Het is heel belangrijk voor elk bedrijf om de verkoop op een bepaald punt te kunnen voorspellen. Deze info wordt gebruikt voor het plannen van een stock, het bepalen van het aantal VTE's, het bepalen van de hoeveelheid basisstoffen die moeten besteld worden bij leveranciers, het berekenen hoeveel transport er nodig is, …

De verkoop in een bepaalde periode kan afhangen van de hoeveelheid verkoop in de vorige week, het budget gespendeerd aan marketing, is het vakantie of niet, …

Met regressie zoeken we een functie die het verband geeft tussen de afhankelijke variabele en onafhankelijke variabelen.

**Vb: gezichtsherkenning**

Wanneer je naar een foto van een gezicht kijk, herken je ogen, neus, mond, … We willen ook een computer leren om een foto van een persoon te herkennen. Hoe leren we een computer waar de ogen, neus, mond van een persoon op een foto zijn? Wanneer we dit kunnen, kunnen er virtuele kleedkamers gemaakt worden waar je met jouw gezicht bvb een zonnebril kan passen en zien hoe de ene of de andere jou staat.

We zoeken bvb de x – en y – waarden van het linkeroog, rechteroog, … Deze posities hangen bvb af van de relatieve posities in de foto en de karakteristieken van de omliggende pixels van een oog in deze foto. De positie van elk punt kan gevonden worden door een aantal afzonderlijke regressieproblemen.

## CLASSIFICATIE VERSUS REGRESSIE

Classificatie en regressie zijn vergelijkbaar op een aantal vlakken.

Herinner de set up van classificatie: problem statement – features – training data – test data

We kunnen regressie op dezelfde manier zien:

1. Definitie van het regressie probleem: bereken een bepaalde continue waarde uit andere variabelen

2. Features: onafhankelijke variabelen. In regressie moeten deze features niet noodzakelijk gerelateerd zijn aan 1 bepaald object

3. Training data: je neemt een dataset waarbij de input **en output** reeds gekend is om het verband te bepalen tussen de onafhankelijke en afhankelijke variabelen ➔ dit maakt dat regressie ook tot "supervised learning" behoort

4. Test fase: een continue waarde wordt toegekend aan de afhankelijke variabele op basis van de onafhankelijke variabele

**Voorbeeld: eenvoudig lineair regressie algoritme onder de loop**

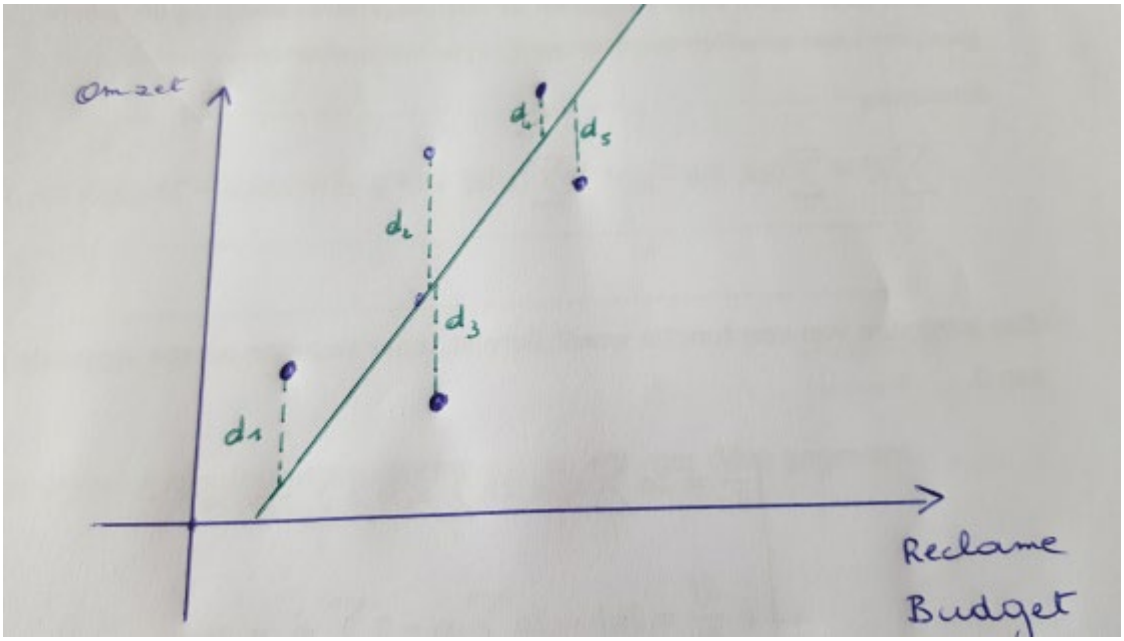
Een belangrijk aspect bij het op de markt brengen van een product is de hoeveelheid geld die aan reclame wordt besteed.  Doel is natuurlijk dat reclame maken ook de verkoop van je product stimuleert.

Aan de hand van gegevens over vroegere verkopen en reclamebudgetten kan je met behulp van regressieanalyse het verband tussen deze twee gegevens nagaan.

**Regressie**: we zoeken een verband tussen een afhankelijke variabele en één of meerdere onafhankelijke variabelen

**Lineaire regressie**: het verband tussen de afhankelijke en onafhankelijke variabelen is lineair

**Eenvoudige lineaire regressie**: er is slechts één onafhankelijke variabele



De regressierechte heeft als vergelijking: $y = ax + b$ met

y = afhankelijke variabele

x = onafhankelijke variabele

a = helling van de regressierechte (richtingscoëfficiënt)

b = snijpunt van de regressierechte met de y – as

Doel: a en b zoeken aan de hand van de gegevens; zodanig dat de rechte de gegevens zo goed mogelijk benadert. Het bepalen van a en b gebeurt hier via de methode van de kleinste kwadraten.

De lineaire regressierechte benadert de 'puntenwolk' nooit exact. Er wordt per gegeven een fout gemaakt. We trachten de totale fout te minimaliseren.

We berekenen $\sum_i d_i$, maar aangezien de ene (negatieve) afwijking de andere (positieve) kan opheffen trachten we $\sum_i d_i^2$te minimaliseren.

Berekening

$$\sum_i d_i^2 = \sum_i (ax_i + b - y_i)^2 = \sum_i (a^2x_i^2 + b^2 + y_i^2 + 2ax_ib - 2ax_iy_i - 2by_i)$$

Een minimum van een functie wordt bereikt, daar waar de eerste afgeleide gelijk is aan 0.

$$\begin{cases} \frac{\delta f}{\delta a} = 2a\sum_i x_i^2 + 2b\sum_i x_i - 2\sum_i x_i\, y_i = 0 \\ \frac{\delta f}{\delta b} = 2nb + 2a\sum_i x_i - 2\sum_i y_i \;=\; 0 \end{cases}$$

Dit geeft volgend stelsel dat opgelost dient te worden:

$$\begin{cases} \sum_i x_i \;\; a + n\;\; b = \sum_i y_i \\ \sum_i x_i^2 \;\; a + \sum_i x_i \;\; b = \sum_i x_i\, y_i \end{cases}$$

Met als oplossing:

$$a = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{\sum x_i^2 - n\bar{x}^2} \quad \text{en} \quad b = \bar{y} - a\bar{x}$$

**Voorbeeld**

Een winkel in huishoudelijke artikelen voert een vijf maandelijks experiment uit om het effect van reclame op de omzet te bepalen. De resultaten vind je hieronder.

| Maand | Uitgaven reclame x (in 10000) | Omzet y (in 10000) |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 2 | 1 |
| 3 | 3 | 2 |
| 4 | 4 | 2 |
| 5 | 5 | 4 |

Zoek een kleinste kwadratenbenadering $y = ax + b$ voor deze gegevens.

## Bekijk PLURALSIGHT: How to think about Machine Learning Algorithms:

## 6. Solving Regression Problems

## - Minimizing Error Using Stochastic Gradient Descent (5 min)

## - Implementing Linear Regression in Python (3 minuten)

# 7.3 Unsupervised ML: Clustering

Clustering is een volgend typisch ML probleem. In deze sectie bespreken we clustering zodanig dat we ook dit type kunnen herkennen in “real life” problemen. We gaan ook even in op het verschil tussen classification en clustering.

We voeren één type van ML clustering algoritme uit nl het K – means clustering.

**Herinner:** Clustering is een manier om items te groeperen op basis van “bepaalde maatstaven van gelijkenissen”. We hebben een grote set van objecten en willen deze indelen in groepen zodat objecten die tot dezelfde groep behoren, gelijkenissen vertonen.

**Voorbeeld**: we willen gedrag van gebruikers van social media begrijpen. Het doel is deze grote groep personen in te delen in kleinere groepen op basis van gelijkenissen. Het basisidee om dit te doen is als volgt:

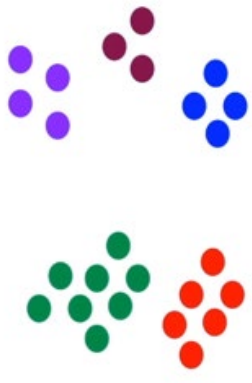
- gebruikers die in dezelfde cluster zitten moeten gelijkenissen hebben (dus de intra cluster gelijkenissen worden gemaximaliseerd)
- gebruikers in verschillende clusters zijn verschillend tov elkaar (dus de inter cluster gelijkenissen worden geminimaliseerd).

De op deze manier gevormde clusters bestuderen, levert vaak interessante informatie op wat betreft gelijkenissen binnen de clusters. Deze kennis kan dan gebruikt worden om gerichte acties te voorzien per cluster (promoties voorstellen, producten aanprijzen, …).

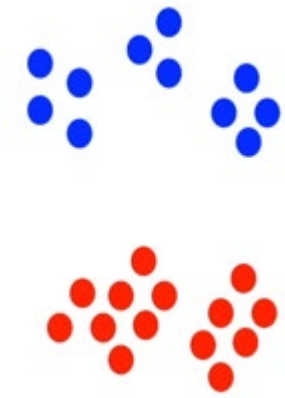
**Werkwijze:**

Alle gebruikers (los van clusters) kunnen voorgesteld worden op basis van bepaalde kenmerken: leeftijd, woonplaats, frequentie in het gebruik van bepaalde topics (of voorkeursscore of keywoord dat gebruikt wordt op social media). Op deze manier kunnen we elke gebruiker voorstellen door een N – tupel van cijfers, dus elke gebruiker wordt voorgesteld door een punt in een N – dimensionale ruimte. Het clustering algoritme gaat groepen zoeken die dicht bij mekaar liggen.

Er zijn meerdere manieren om de puntenwolk op te delen in clusters:



5 clusters



2 clusters

Elke cluster heeft sommige gemeenschappelijke kenmerken maar wij weten nog niet welke dit zijn.

Visueel is het duidelijk wat er gebeurt. In beide manieren zijn de clusters gebaseerd op de afstand tussen gebruikers: we minimaliseren de gemiddelde afstand tussen gebruikers in één en dezelfde cluster en maximaliseren de afstand tussen gebruikers die zich in verschillende clusters bevinden.

Afstand is een abstract begrip in deze N – dimensionale ruimte, maar in "real life" kan deze afstand betekenisvol zijn vb. geografisch op dezelfde plaats wonen, voorkeur hebben voor dezelfde dingen, zelfde leeftijd hebben of …

Typische set – up van clustering problemen:

- Dataset: de volledige dataset (alle items) die gegroepeerd moet worden

- Features: elk item in de dataset wordt voorgesteld in numerieke waarden. Gebruik features die relevant zijn voor de items en patronen/groepen waar jij naartoe werkt.

- Clustering algoritme: gebruik een clustering algoritme om de items te clusteren

**Voorbeeld van Features:**

1. je wil gebruikers classificeren op basis van gebruikspatronen: zijn er misschien groepen te herkennen in routines omtrent het tijdstip wanneer men inlogt op social media? Dan kan je gegevens gebruiken als
    - Hoe vaak logt men ’s morgens in?
    - Hoe vaak logt men ’s avonds in?
    - Hoelang blijft een gebruiker ingelogd?

2. je wil gebruikers classificeren op basis van hun interesses of likes op social media. Je maakt een lijst van 100 topics en dan kan je gegevens gebruiken als
    - Aantal likes van elk topic
    - Aantal shares van elk topic

**Bestaande clustering Algoritmes** zijn o.a. K – means clustering, hiërarchical clustring, density-based clustering, distribution-based clustering

**Verschil classificatie – Clustering**

Classificatie: data classificeren in **voor gedefinieerde** categorieën. Je neemt één object en probeert een label toe te kennen aan dit object.

Clustering: data groeperen in een set van **categorieën** (deze zijn op voorhand **niet gekend).** Bij clustering deel je een grote groep op in niet gekende categorieën.

Er zijn verschillen tussen classificatie en clustering maar vaak gaan ze ook hand in hand. Stel: je wil een aantal artikels indelen in groepen (clusters) gebaseerd op thema’s. Wanneer de artikels ingedeeld zijn in groepen mbv clustering, kan je via classificatie een nieuw artikel toekennen aan één of andere groep.

**Clustering toegepast: K – Means algoritme voor het clusteren van documenten**

Er bestaan verschillende soorten documenten: artikels, reviews, tweets, boeken, webpagina's, …

Gegeven een bepaalde set van documenten, groepeer (cluster) deze op basis van gelijkenis. Na de clustering ontdek je misschien dat er interessante thema's zijn binnen de clusters: artikels behorende tot dezelfde cluster, zijn van dezelfde auteur of zijn van hetzelfde uitgiftejaar of hebben een vergelijkbaar onderwerp of…

1. Dataset: al de documenten die je gebruikt om de clustering uit te voeren

2. Features: er zijn meerdere manieren om "tekst" om te zetten in numerieke waarden. Een veel gebruikte is "term frequency representation" (zie eerder) om documenten om te zetten in numerieke waarden

   Vb: we hebben alle woorden die voorkomen in alle teksten in een tupel voorgesteld.

   $$(hallo, dit, zijn, alle, woorden, en, leestekens, die, ook, bestaan, in, teksten)$$

   Elke tekst kunnen we nu in cijfers in een tupel voorstellen door te tellen hoe vaak een woord voorkomt in die tekst.

   $$(dit, zijn, teksten, en, dit, ook)$$

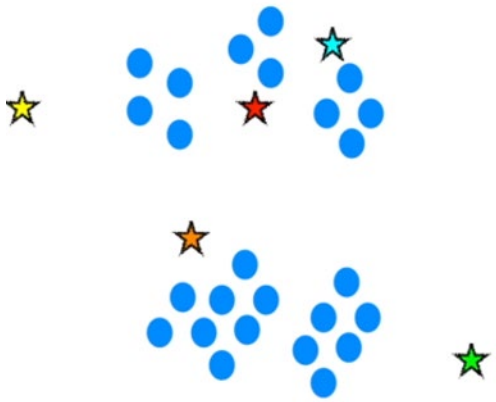
   $$(0,2,1,0,0,1,0,0,1,0,0,1)$$

   Merk op: niet elk woord in een tekst draagt evenveel bij tot de betekenis van deze tekst zoals: het, de, een, en, of, van, …. Bij TF – IDF wordt hier rekening gehouden door gewichten mee te geven aan elk woord. Dit in tegenstelling tot TF waar ieder woord een gelijk gewicht krijgt.
   Woorden die **minder frequent** voorkomen, bepalen **meer** de inhoud van zinnen/tekst. Dus: woorden die **vaak** voorkomen, willen we **een lager gewicht** meegeven en woorden die **minder vaak** voorkomen een **hoger gewicht**. Dit is TF – IDF: term frequency – inverse document frequency (omgekeerde document frequentie) met als mogelijk gewicht:

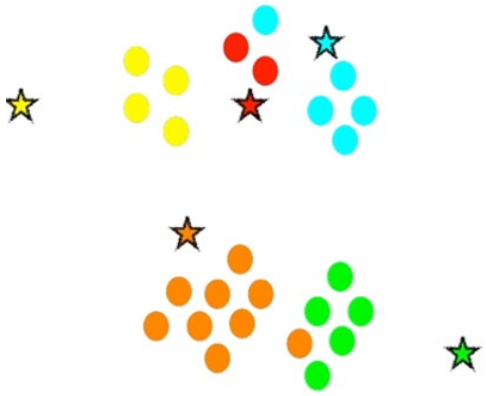
   $$gewicht = \frac{1}{\#\ documenten\ waar\ dit\ woord\ in\ voorkomt}$$

3. K – means clustering: dit algoritme gaat onze data indelen in K groepen waarbij K op voorhand bepaald wordt door de gebruiker.
   Alle documenten zijn voorgesteld gebruik makend van TF – IDF.
   Elk document is een N - tupel van cijfers waarbij N het totaal aantal verschillende woorden is in alle documenten samen. Op deze manier wordt elk document een punt in een N – dimensionale ruimte en kunnen dit dus uitzetten in een N dimensionaal assenstelsel.

   *STAP 1*: We beginnen met het initialiseren van een set van middens (de K – means = K middens van de clusters die we willen vinden). Het aantal clusters wordt vastgelegd door de gebruiker (zeg 5). De initiële middens kunnen random zijn maar er zijn technieken om deze initiële waarden op een zodanige manier te bepalen dat het algoritme sneller tot convergeren overgaat (hier gaan we niet verder op in).

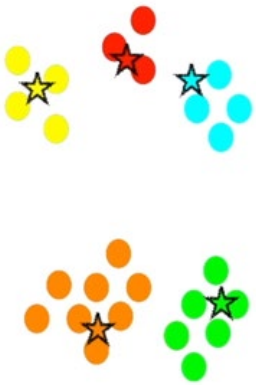


STAP 2: Nadat de initiële middens gekozen zijn, zal elk punt (dus elk document) toegewezen worden aan één van deze initiële middens. Op deze manier ontstaan de eerste clusters.



STAP 3: Eens deze clusters gemaakt zijn, zoeken we nieuwe middens (die onze initiële middens vervangen) door opnieuw het midden van elke cluster te zoeken / bepalen.
We herhalen stap 2 en 3 totdat de nieuwe middens niet meer wijzigen.

Dit noemen we convergentie. En op deze manier hebben we de K (=5) clusters waarin we de documenten wilden verdelen.



## PLURALSIGHT: How to think about Machine Learning Algorithms:

## 4. Clustering Large Data Sets into Meaningful Groups – Implementing K – Means Clustering (5 minuten)

## 7.4 Unsupervised: Recommendations

Gepersonaliseerde “aanbevelingen” winnen aan belang. Ze hebben een (positieve) invloed op oa. koopgedrag. Het kunnen voorspellen van gebruikerstevredenheid is hier nauw mee verbonden. We bekijken dit nader met een techniek die “Collaborative filtering” heet.

Recommendation wordt “overal” toegepast:

- bezoek eender welke “commerciële” website en je krijgt reclame over producten die je misschien wel leuk vindt en eventueel zal gaan aankopen.
- Wanneer je Netflix of Spotify of… gebruikt, krijg je ook aanbevelingen voor programma’s, films, liedjes die je misschien leuk vindt gebaseerd op hetgeen je reeds bekeken, geluisterd hebt.

Dit zijn voorbeelden van gepersonaliseerde aanbevelingen. Deze diensten voorspellen wat je nodig hebt of graag hebt, soms zelfs nog voor je het zelf beseft.

Dit is een erg grote business geworden, zeker in de online wereld.

Vroeger waren webpagina’s statisch (vergelijkbaar met een fysieke winkel). Reclame die verscheen, het uitzicht van een website, … was gemaakt voor het “doorsnee” publiek; dus niet gepersonaliseerd. Dit is geëvolueerd naar pagina’s waar (te pas maar soms ook te onpas) gepersonaliseerde boodschappen verschijnen.

Stel: een winkel kan veranderen afhankelijk van de klant die binnenkomt en enkel de producten die hij/zij nodig heeft, worden voor hem/haar getoond (je moet dus niet meer de hele winkel rond om alles te kopen wat jij nodig hebt). Dit is natuurlijk niet mogelijk in de praktijk bij reële, fysieke winkels maar WEL voor online shops.

De online shop kan gepersonaliseerd worden op basis van voorkeur, noden en behoeften van een klant.

Online winkels hebben standaard zeer uitgebreide catalogi: Netflix (zeer uitgebreid aanbod aan films), Spotify (zeer ruim aanbod aan liedjes), Amazon … maar gebruikers beperken zich vaak tot het bekijken van enkel de startpagina of een paar andere pagina's die maar een paar muisklikken verwijderd zijn. Gebruikers hebben dus sturing nodig in het vinden van wat ze zoeken of zelfs in wat ze interessant vinden. Soms kunnen aanbevelingen gebruikers leiden tot iets waarvan ze eigenlijk nog niet wisten dat ze het nodig hadden of leuk vinden.

Iets anders waar recommendations toe bijdraagt is "trouw zijn aan" een website: hoe langer je een bezoeker kan houden bij jouw site, hoe meer geneigd hij/zij is om te kopen, wat het uiteindelijke doel is van een webwinkel. Dus zoveel te persoonlijker je een website kan maken voor de klant, zoveel te meer geneigd de klant is om veel tijd te spenderen op deze site en dus meer te kopen.

Dit is waar deze tak van ML om draait: de klant **nieuwe noden laten ontdekken** en de klant **trouw laten blijven** aan eerdere producten.

Een aantal (online) voorbeelden van ML:

**Voorbeeld**: zoek de top 10 films die interessant zijn voor een gebruiker

- Bereken een score voor elke film op basis van ratings gegeven door alle gebruikers
- Sorteer de films op basis van deze ratings
- Neem hieruit de top 10 films die de gebruiker nog niet gezien heeft

**Voorbeeld**: aanbevelingen op basis van browser geschiedenis

- Gebruik bvb het aantal keren dat een product bekeken is als een score voor dat product
- Bereken de score van alle producten op basis van de views door alle gebruikers
- Selecteer hieruit de top 10 producten voor een specifieke user

**Voorbeeld:** wanneer je een product koopt, krijg je vaak meldingen in de aard van: andere bezoekers kochten ook dit…

- We berekenen voor elk product een score
- Hieruit zoeken we naar producten met een hoge score onder de gebruikers die reeds het product waar we ons op focussen, gekocht hebben. We zoeken dus naar een deelverzameling van producten met een hoge rating onder users die het specifieke product al gekocht hebben
- Selecteer hieruit de top 10 producten

Het gemeenschappelijk probleem in voorgaande voorbeelden is dat we een score moeten berekenen voor alle producten gebaseerd op historische scores, gegevens, muisklikken, … van alle gebruikers.

Dit wil zeggen dat de rating van een product voorspeld moet worden voor een bepaalde gebruiker zelfs als die gebruiker nog geen indicatie gegeven heeft dat hij/zij geïnteresseerd is in dit product. Hiervoor gebruikt men "Collaborative Filtering": deze algoritmes voorspellen scores van producten die een gebruiker nog niet gekocht of gezien heeft op basis van gedrag uit het verleden van andere vergelijkbare gebruikers (cfr definitie Collaborative Filtering)

De **basisaanname van Collaborative Filtering** is zeer simpel: als je 2 gebruikers vindt met dezelfde mening over heel wat producten, dan zullen ze ook dezelfde mening hebben over andere producten.

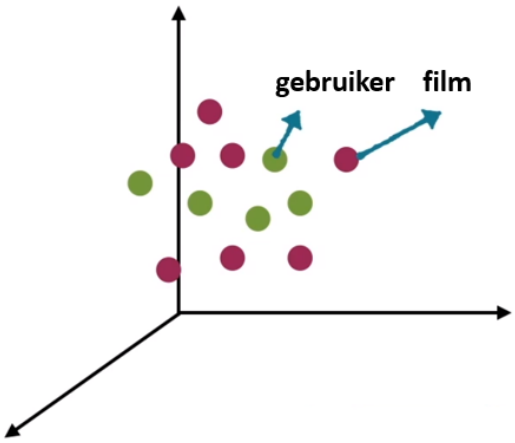
**Voorbeeld:** Voorspel de rating van een film: hiervoor is er nood aan training data:

- gebruikersnummer
- productnummer (boek, video, film, artiest, artikels, woorden in e-mails)
- rating (= om het even welke maatstaf die een gebruikersvoorkeur weergeeft). Dit kan een expliciete rating zijn (de user heeft ergens echt zijn mening, score aan gegeven; bv door een enquête in te vullen) maar het kunnen ook impliciete ratings zijn (bepaald op basis van aantal keer geklikt op dit item, aantal aankopen, aantal keer gedeeld, aantal likes, …). Onderliggend zijn er uiteraard factoren die de rating van een film beïnvloeden o.a.
    - genre van de film
    - cast van de film
    - tijdstip van release
    - geslacht van de user

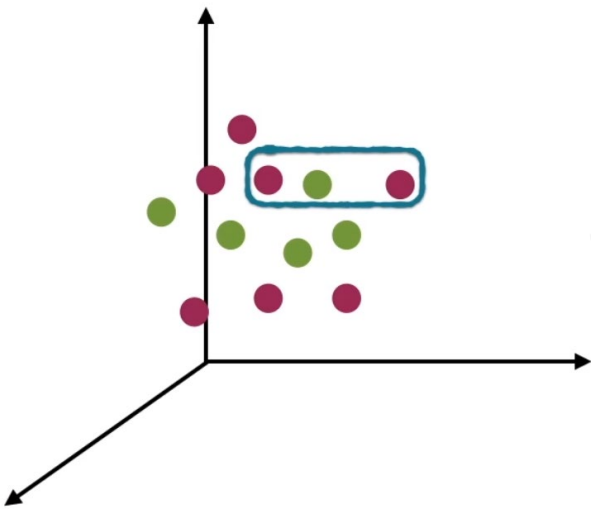
Een werkwijze om dit probleem aan te pakken kan zijn:

- geef een rating aan elke gebruiker op basis van hun scores op deze factoren (het belang dat zij hechten aan elk van deze factoren) (vb. getal van 1 tot 5)
- geef analoog een rating aan elke film op deze factoren (vb. getal van 1 tot 5)

Hierdoor kan elke gebruiker en elke film voorgesteld worden door 4 scores (ratings). Dit kunnen we ons voorstellen als punten in een 4 – dimensionale ruimte:




We bekijken nu gebruiker per gebruiker en zoeken naar de films die het “dichtst” staan bij deze gebruiker. Deze films zouden de films kunnen zijn waar de gebruiker een voorkeur voor heeft.



Deze methode is een logisch en intuïtief vrij snel te begrijpen algoritme maar de complexiteit zit in volgende aannames:

- we moeten de verborgen factoren kennen (genre, cast, …)
- we moeten deze factoren kunnen kwantificeren voor de gebruikers (1 – 5)
- we moeten deze factoren kunnen kwantificeren voor de films (1 – 5)

Deze gegevens zullen zelden mooi in een dataset samengevat zijn waardoor bovenstaande idee praktisch vaak niet uit te voeren valt.

Vandaar dat vaak (eerst) toegespitst wordt op een deelverzameling van Collaboratorive Filtering nl Latent Factor Analysis: Latent Factor Analysis probeert deze verborgen factoren die invloed hebben op gedrag, te identificeren. Latent Factor Analysis gebruikt in het geval van de films enkel gebruikers gegevens, product gegevens en ratings om verborgen factoren te identificeren. De factoren die uit Latent Factor Analysis komen kunnen betekenisvol zijn in het dagelijkse leven (bvb genre, cast van de film, …) maar het kunnen even goed abstracte factoren zijn zonder enige betekenis.

**Werking van Latent Factor Analysis:**

Stel de training data (gebruikers, producten (= films), ratings) voor door een matrix (zeg R) waarbij de rijen de gebruikers voorstellen en de kolommen de producten. De waarden zijn de ratings van een specifieke gebruiker voor een specifiek product.

| | Prod 1 | Prod 2 | Prod 3 | … | | Prod D |
|---|---|---|---|---|---|---|
| Gebr 1 | 4 | - | 4 | - | - | 2 |
| Gebr 2 | - | - | 2 | 1 | - | - |
| Gebr 3 | 2 | 1 | - | - | 5 | - |
| … | 1 | - | 1 | - | - | - |
| … | - | 2 | - | 5 | 2 | 3 |
| Gebr N | 3 | 4 | - | - | 1 | - |

Latent Factor Analysis schrijft bovenstaande matrix als een product van 2 aparte matrices nl een gebruikers-matrix waarbij elke rij één gebruiker voorstelt (zeg G):

| | Factor 1 | Factor 2 | Factor 3 |
|---|---|---|---|
| Gebr 1 | | | |
| Gebr2 | | | |
| Gebr 3 | | | |
| … | | | |
| … | | | |
| Gebr N | | | |

en een product-matrix waarbij elke kolom één product voorstelt (zeg P):

| | Prod 1 | Prod 2 | Prod 3 | … | … | Prod D |
|---|---|---|---|---|---|---|
| Factor 1 | | | | | | |
| Factor 2 | | | | | | |
| Factor 3 | | | | | | |

zodat R = G . P (matrixproduct) en Factor 1 tot en met Factor 3 de verborgen factoren zijn. Dus elk cijfer in R is het matrix product van één rij uit G en één kolom uit P.

vb.: de rating van gebruiker 4 voor product 3 (nl 1) = $R_{43} = G_4 \,.\, P_3^T$ (matrixproduct)

Belangrijk hierbij is op te merken dat initieel heel wat cellen in de rating – matrix (R) leeg zijn. Wanneer we **elke** cel in de hele rating - matrix schrijven als een vergelijking en dus een stelsel van bovenstaande vergelijkingen oplossen, kunnen we alle ontbrekende cellen in matrix R berekenen.

Het probleem vertaalt zich dus naar

- zoek per gebruiker de factor waarden (rijen van G)
- zoek per product de factor waarden (kolommen van P)

zodat de totale fout geminimaliseerd wordt:

$$min_{G\,\&\,P} \sum (R_{ui} - G_u P_i^T)^{\,2}$$

Op deze manier hebben we ons probleem omgevormd tot een minimalisatie – probleem. Een veel gebruikt algoritme om deze minimalisatie uit te voeren is het "Alternating Least Squares Algoritme".

**Hoe werkt het Alternating Least Squares algoritme?**

We concentreren ons op één bepaalde rating: bv. $R_{ui}$ . De vergelijking die opgelost dient te worden is: $(R_{ui} - G_u P_i^T)^{\,2} = 0$. Dit is één kwadratische vergelijking waarin 2 onbekenden zitten nl $(G_u \;en\; P_i)$. Hoe lost het Alternating Least Squares algoritme het probleem van 1 vergelijking met 2 onbekenden op?

1. $G_u$ krijgt een vaste waarde waardoor de vergelijking opgelost kan worden naar de overblijvende variabele $P_i$
2. Deze waarde voor $P_i$ wordt in de vergelijking gestopt en dan opgelost naar de overblijvende variabele $G_u$
3. Stap 1 en 2 worden herhaald totdat $G_u$ en $P_i$ niet meer wijzigen (dit noemen we convergentie)

Uit dit algoritme vinden we nu de waarden in bovenstaande matrices. Het afleiden van betekenisvolle factoren hieruit is geen sinecure.

**Een eenvoudig voorbeeld ter verduidelijking van het inhoudelijke aspect …**

Stel dat een groep proefpersonen van een vragenlijst twee vragen, A en B, beantwoordt. Uit analyse blijkt dat er een verband is tussen A en B. Dan kan dit komen door de invloed van A op B, door de invloed van B op A, of doordat er nog een andere onbekende variabele C in het spel is. Met factoranalyse kan de onbekende variabele C opgespoord worden.

Als voorbeeld nemen we een proef waarbij van een aantal personen de lengte van de armen (A) en de lengte van de benen (B) worden gemeten. Deze blijken goed gecorreleerd te zijn. De onbekende factor zou hier de grootte van de persoon kunnen zijn (C).

Zoals ook de andere ML algoritmes, kan je het Aternating Least Squares algoritme terugvinden in voorgedefinieerde libraries van bvb Python.

Wat we wel moeten meegeven aan het algoritme is het aantal verborgen factoren waar je naar op zoek bent. Zoveel te groter je dit aantal neemt, zoveel te complexer wordt het ML algoritme. Het uitgangspunt van (elk) ML algoritme is om zo eenvoudig mogelijke modellen te vinden. Wanneer er een keuze is tussen een model met 10 of 1000 factoren en de error is niet noemenswaardig groter met 10 factoren, dan wordt standaard dit model gekozen. Dit wordt in acht genomen in het algoritme onder de vorm van een Lambda (een straf voor het kiezen van teveel factoren). Deze Lambda moet de gebruiker ook meegeven als parameter aan het ML algoritme. Het aantal factoren en de waarde van Lambda is iets waar je mee kan spelen; die je via trial and error kan bepalen.

## Bekijk PLURALSIGHT: How to think about Machine Learning Algorithms:

## 7. Recommending Relevant Products to a user - Implementing ALS to find Movie Recommendations (3 minuten)

# 8. Bronnen

**PluralSight:**


- Understanding Machine Learning (David Chapell)
- Understanding Machine Learning with Python (Jerry Kuratta)
- How to think about Machine Learning Algorithmes (Swetha Kolalapudi)
- Play by Play: Machine Learning Exposed (Katharine Beaumont & James Weaver)


**Websites:**


- http://www.cad.zju.edu.cn/home/zhx/csmath/lib/exe/fetch.php?media=2011:presentation_ml_by_ibrar.pdf
- https://towardsdatascience.com/the-5-clustering-algorithms-data-scientists-need-to-know-a36d136ef68
- https://www.python-course.eu/machine_learning.php
- https://internetofthingsnederland.nl/wat-is-machine-learning/
- https://www.youtube.com/watch?v=GGWBMg0i9d4
- https://3bplus.nl/wat-is-machine-learning-een-introductie/