HOGESCHOOL PXL

PXL-DIGITAL

PXL - IT

Data Advanced Deel 3: MACHINE LEARNING

Lector(en)

Isabelle Godfrind

Kerstin Nys

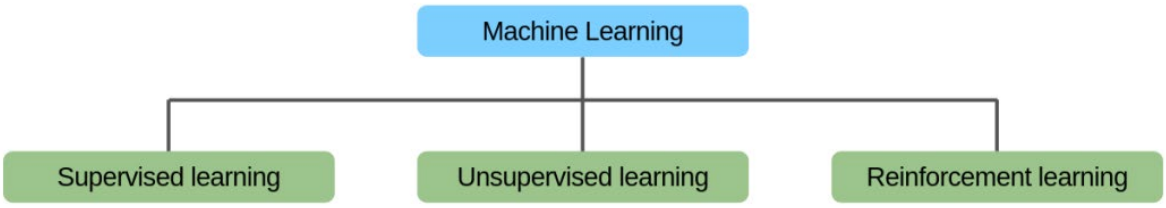
Heidi Tans

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.

Er zijn 3 soorten waarop een machine kan leren:



In deze introductie nemen we 2 soorten Machine Learning onder de loep: Supervised en Unsupervised. Van elke soort bestuderen we onderstaande categorieën:

- Supervised Machine Learning
    - Classificatie
    - Regressie
- Unsupervised Machine Learning
    - Clustering

Verder gaan we in op Recommendations: algoritmes die relevante aanbevelingen doen aan gebruikers.

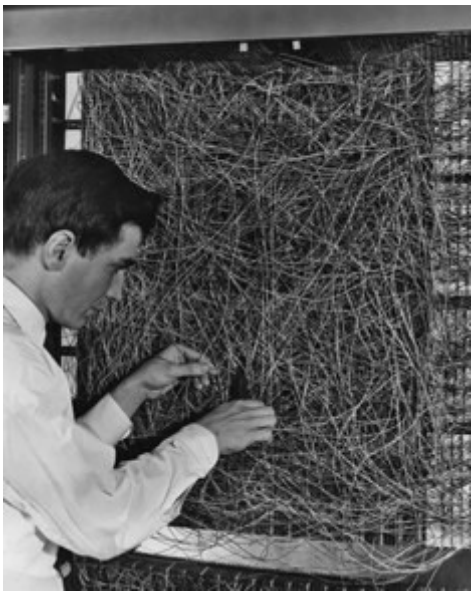
2. Historiek

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

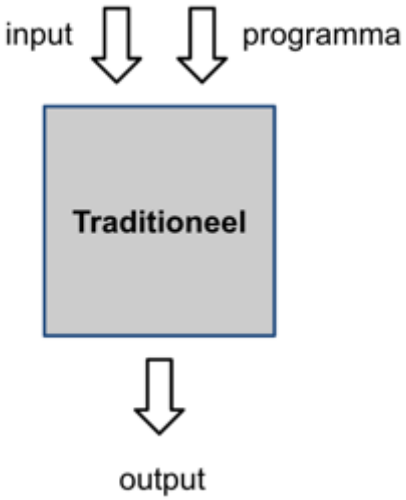
Je bent een “alien” en kijkt naar de bevolking ergens op de wereld en je weet dat er 2 soorten mensen op de wereld zijn: zeg man en vrouw. Maar je kent de karakteristieken om te differentiëren tussen man en vrouw niet.

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"
```

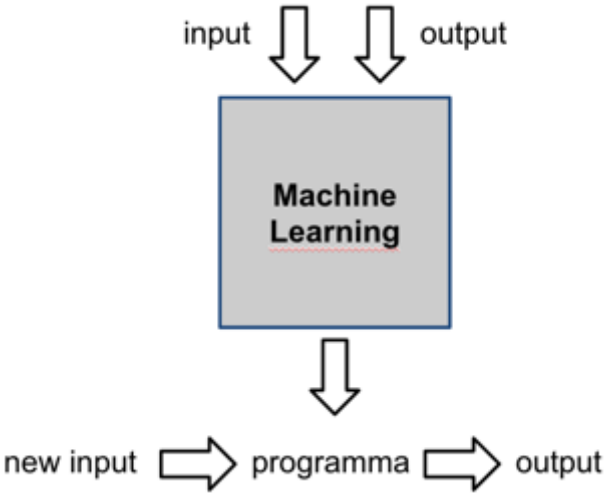
Je steekt data én een programma in een computer en je verkrijgt output; dit is een benadering van het probleem op basis van regels; op basis van programmatielogica.



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!

Wat is het verschil met een Rule based aanpak?

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

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.

Benadering mbv ML: Deze benadering bestaat uit 3 stappen:

1. Verzamel een grote set data ivm mensen (10 of zelfs 100 verschillende variabelen zouden mensen kunnen typeren) en blijf deze ook verzamelen op een continue manier.
2. Gebruik een algoritme dat "zelfstandig" het verband vindt tussen deze variabelen en het geslacht van een mens
3. Update dit verband voortdurend mbv nieuwe 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.

- 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.

- 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.

- 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.

Wanneer gebruik je ML dan? Enkele vuistregels:

- Wanneer 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.

4. 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

4.1 Bepaal welk type ML probleem we hebben

4.2 Gebruik de data die je hebt

4.3 Pas een standaard algoritme toe op jouw data

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

4.1 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. Bvb we willen een persoon categoriseren als “male” of “female”. De data die we gebruiken om een label te plakken op deze persoon bevat een aantal personen en hun karakteristieken maar **ook het juiste label** (juiste outcome: male of female) is aanwezig.

In meer wiskundige bewoordingen:

- De inputvariabelen $(x_1, x_2, \dots, x_n)$ alsook de outcome variabele *y* is aanwezig in de training data set
- Het algoritme is getraind om de functie f te bepalen die het verband tussen beide geeft nl $y = f(x_1, x_2, \dots, x_n)$
- Mbv deze functie voorspellen we op basis van een nieuwe income $(x_1, x_2, \dots, x_n)$ de outcome *y*
- Verder wordt de functie f continu bijgestuurd

Bij **Unsupervised Machine Learning** beschikt het algoritme over training data die enkel voorzien is van input (geen output).

Voorbeeld

Netflix die jou een nieuwe serie aanraadt. Hoe gaat dit in zijn werk? Netflix plaatst jou in een groepje (cluster: vooraf niet gedefinieerd) met andere gebruikers die ongeveer dezelfde series als jij bekeken. Deze gebruikers hebben die nieuwe serie ook bekeken en positief beoordeeld.

In meer wiskundige bewoordingen:

- Enkel de inputvariabele $(x_1, x_2, \ldots, x_n)$ is aanwezig in de training data set (geen outcome *y*)
- Het algoritme brengt patronen, structuren, groepen, … naar voren die op voorhand niet opgenomen zijn in de dataset.

4.2 Data

Wanneer je voor ML gekozen hebt, heb je een grote hoeveelheid historische data nodig. Deze data kan numeriek van aard zijn maar tevens beschikbaar zijn in ongestructureerde tekst, video's, foto's, … Om deze data te gebruiken in een ML context is het nodig 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.

4.3 ML - algoritme

Tot slot gebruik je een algoritme om in de verzamelde data (historische data) 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.

Welke algoritme er gekozen wordt, hangt af van het type probleem:

In tegenstelling tot wat doorgaans gedacht wordt, is het uitvoeren van het algoritme niet het meest bepalende deel in het oplossen van ML probleem aangezien heel wat programmeertalen "build in packages" hebben waarin deze algoritmes voorzien zijn. Er bestaan vuistregels voor het beslissen wanneer welk algoritme toe te passen. Je kan ook via "plug and play" meerdere bestaande algoritmes uitvoeren op jouw data. De setup van het probleem, het klaarmaken van de data en het juist presenteren van data en output, vergt vaak het meeste energie.

5. ML problemen uitgewerkt

In volgende paragrafen gaan we eerst dieper in op Supervised Machine Learning. Hier wordt zowel Classificatie (een observatie wordt toegekend aan één of andere categorie) als Regressie (een continue waarde wordt voorspeld) onder de loep genomen.

Bij Unsupervised Machine Learning behandelen we Clustering (items worden gegroepeerd op basis van gelijkenissen).

We sluiten af met een blik op Recommendations.

5.1 Supervised ML: Classificatie

Alle classificatie problemen hebben een aantal gemeenschappelijke karakteristieken:

- Doel: Een object (= problem instance, nieuwe observatie) toekennen aan één of andere categorie (=classificeren)

- De categorieën (labels) waaraan we een object toekennen, zijn op voorhand gekend. We onderscheiden verschillende soorten classification:
    - binary classification: outcome kan slechts uit 2 categorieën bestaan bvb Male of Female – Spam of geen Spam – Reclame of niet
    - multi-class classification: outcome kan uit meerdere categorieën bestaan bvb Setosa, Versicolor of Virginica (Iris data set)
    - multi-label classification: per object (instance) worden meerdere labels voorspeld bvb (True female) of (True Male) of (False Female) of (False Male)
    - multi-output classification: multi-class + multi-label samen bvb (zondag, januari)

- Classifier is een algoritme dat de indeling uitvoert en een label geeft aan het object. Beschouw de classifier voorlopig als een black box. Wat binnen in die black box gebeurt, kunnen we ons voorstellen als wiskundige of statistische algoritmes (regels of vergelijkingen).
  Voorbeelden van classifiers: Naive Bayes Algoritme, Support Vector Machine Algoritme, Decision Tree algoritme, Logistic Regression algoritme, Linear Discriminant Analysis algortime, K-nearest neighbor, Random Forests, Logistic Regression, …

- Features (eigenschappen – kolommen in onze dataset): We moeten van onze data features verzamelen die representatief zijn voor het probleem dat je wil oplossen. Belangrijk is dat deze features numeriek van aard zijn. Zelfs als je tekst of foto's gebruikt als data voor je ML probleem, moet je deze gegevens omzetten naar een numerieke grootheid.

- Training: De classifier gebruikt (een deel van de) data (training data) waarvan de categorie reeds gekend is (Supervised ML). Van deze training data leert onze classifier hoe nieuwe data te classificeren. Dit noemt men "het model trainen". Het tracht regels en patronen af te leiden van de training data. De training data kunnen we zien als een tupel van features én label.

- Testing: test het model gebruik makend van test data. Soms bestaat deze testdata uit een deel van de originele dataset die niet gebruikt is in de training fase. Indien er niet genoeg data voorhanden is, wordt nieuwe data gebruikt als testdata. In deze fase classificeren we de testdata 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.

In onderstaande secties bespreken we 6 bestaande classificatiealgoritmes. Bij een aantal gaan we dieper in op de wiskundige / statistische achtergrond, andere werken we uit op een dataset in een jupyter notebook.

5.1.1 Naive Bayes algoritme: Aan de hand van 2 voorbeelden (personen classificeren en sentiment analysis) tonen we hoe kansrekening aan de basis ligt van dit algoritme.

5.1.2 Support Vector Machine algoritme: De basis waarop dit algoritme steunt om classificatie te doen, wordt uitgelegd aan de hand van een voorbeeld (afbeelding classificatie)

5.1.3 Decision Tree algoritme: Dit algoritme werken we praktisch uit in een jupyter notebook op de Irisdata set om een bloem te classificeren als Setosa, Versicolor of Virginica.

5.1.4 Logistic Regression algoritme: Dit algoritme werken we praktisch uit in een jupyter notebook op de Titanic dataset om te voorspellen of iemand de ramp overleeft of niet.

Van de laatste twee schetsen we kort het algemene idee achter de algoritmes.

5.1.5 Linear Discriminant Analysis

5.1.6 Nearest Neighbor algoritme

5.1.1 Naive Bayes algoritme

Dit algoritme is gebaseerd op voorwaardelijke kansen en de “regel van Bayes”.

Voorbeeld: Persoon classificatie (politieagent of jogger)

Je loopt op straat iemand voorbij. Is dit een politieagent of een jogger? Deze persoon had handboeien en een badge bij.

Bemerking 1: Je realiseert je dat er vandaag een grote joggingwedstrijd georganiseerd wordt in jouw gemeente dus… er zullen wel meer joggers op straat zijn dan politieagenten. Je observeert de volgende 10 personen die voorbijlopen en telt 9 joggers en 1 politieagent.
Dit levert volgende kansen P(jogger) = 0.9 en P(politieagent) = 0.1

Bemerking 2: specifieke attributen behoren vaker tot de ene dan de andere categorie.
Bv sportschoenen ➔ jogger
Bv handboeien, walkietalkie ➔ politieagent

Je observeert de volgende voorbijkomende personen en merkt volgende aantallen

Attribuut	Politieagent	jogger
Handboeien	6	0
sportschoenen	2	8
Wapen	9	0
badge	8	3
Walkietalkie	8	0

Dit kunnen we voorstellen als voorwaardelijke kansen:

$P(jogger \mid handboeien, badge)$= de kans dat iemand een jogger is gegeven dat deze persoon handboeien en een badge bij heeft

Stap 1: bereken deze kans $P(jogger \mid handboeien, badge)$ via de wet van Bayes

Stap 2: bereken ook de kans dat deze persoon een politieagent is $P(politieagent \mid handboeien, badge)$ via de wet van Bayes

Stap 3: vergelijk deze kansen en we kiezen deze categorie die de hoogste kans heeft.

Voorbeeld: Sentiment Analysis

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 is er onnoemelijk veel data voorhanden. De data bestaat vooral uit tweets, meldingen, emoji's, … Deze data is vrij ongestructureerd maar wel publiekelijk toegankelijk voor iedereen.

Wat doet sentiment analysis: We hebben een commentaar, tweet, .. en we moeten deze classificeren als zijnde ofwel positief ofwel negatief. Features (variabelen, dus onze kolommen) om de data te representeren zijn hier niet zo evident: we hebben ongestructureerde data en voor een classificatieprobleem hebben we nood aan numerieke maatstaven.

Vooraleer aan het effectief classificeren van tweets te kunnen beginnen (op basis van de "regel van Bayes"), moet de ongestructureerde data eerst omgezet worden naar numerieke data. 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 gaan hier eerst even dieper op in.

Hoe werkt **"term frequency representation"**?

Men maakt 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 tupel van getallen. Elk getal stelt het aantal keer voor dat het specifieke woord in het geheel van de training data voorkomt. Dit noemt men de woordfrequentie.

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

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

De testzin *(hallo, dit, is, een, tekst, en, een, tekst)* wordt dan voorgesteld door volgend N-tupel (14-tupel in ons voorbeeld):

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

Nu de “ongestructureerde” tweets omgezet zijn in numerieke waarden mbv term frequency representaiton, kan Naive Bayes zijn werk doen om de tweets als positief / negatief te classificeren.
Dit noemen we de **“training fase”**: hier maakt het algoritme (**Naive Bayes** in ons geval) gebruik van commentaren die al gelabeld zijn als zijnde positief of negatief (de training data set) om nieuwe tweets te classificeren.

Income	**Outcome**
tekst 1 $\xrightarrow{term\ freq\ repr}$ **(1,0,3,1,…,0)**	*Positief*
tekst 2 $\xrightarrow{term\ freq\ repr}$ **(1,1,2,1,…,1)**	*Positief*
tekst 3 $\xrightarrow{term\ freq\ repr}$ **(2,0,1,0,…,0)**	*Negatief*
tekst 4 $\xrightarrow{term\ freq\ repr}$ **(0,0,1,1,…,2)**	*Positief*
tekst 5 $\xrightarrow{term\ freq\ repr}$ **(1,1,1,1,…,0)**	*Negatief*
tekst 6 $\xrightarrow{term\ freq\ repr}$ **(1,2,0,0,…,0)**	*Positief*
…	
tekst 1000 $\xrightarrow{term\ freq\ repr}$ **(1,0,2,2,…,3)**	*Positief*

Mbv de training data berekent het algoritme de kans op een positieve tekst en wel als volgt:

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

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 $1 - P_O = 45\%$

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”:

$$P_{blij} = P(blij|Pos\ tekst) = \frac{Som\ van\ alle\ woordfreq\ van\ blij\ in\ pos\ teksten}{Som\ van\ alle\ woordfreq\ van\ blij\ in\ hele\ training\ data}$$

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

Hiermee zijn we aanbeland in de **testfase.** We passen voorgaande toe op testdata 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"

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

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

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. Het Naive Bayes algoritme veronderstelt dat elke variabele **onafhankelijk** bijdraagt tot de positiviteitsscore.

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

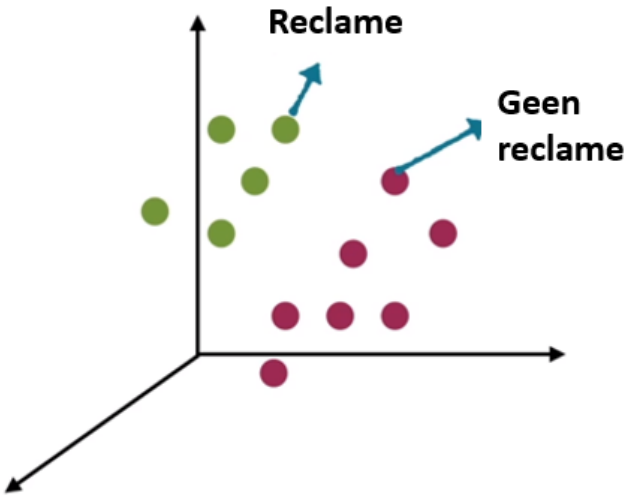
- weinig training data voorhanden hebt
- wanneer je zelf weinig verdere info hebt of kennis in het domein

5.1.2 Support Vector Machine Algoritme

Voorbeeld: add detection

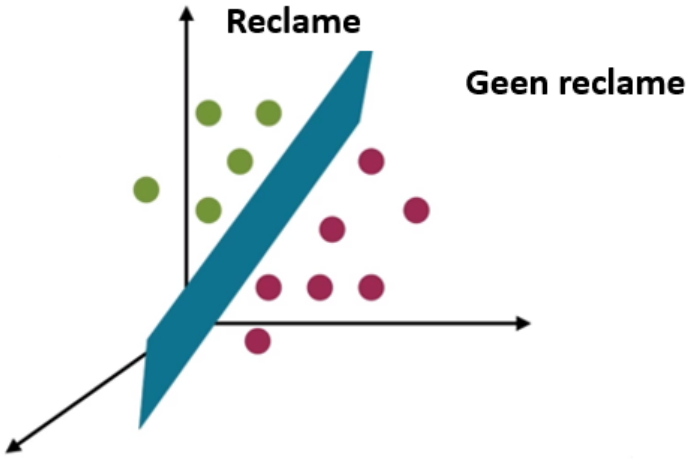
Men wil bepalen of een afbeelding (problem instance) een reclame advertentie is of niet. De features zijn tupels van numerieke waarden van die beelden bvb hoogte, breedte, url, tekst op pagina, tekst bij afbeelding, ...

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 N getallen geassocieerd.
Deze punten kunnen voorgesteld worden in een N – dimensionale figuur.
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.

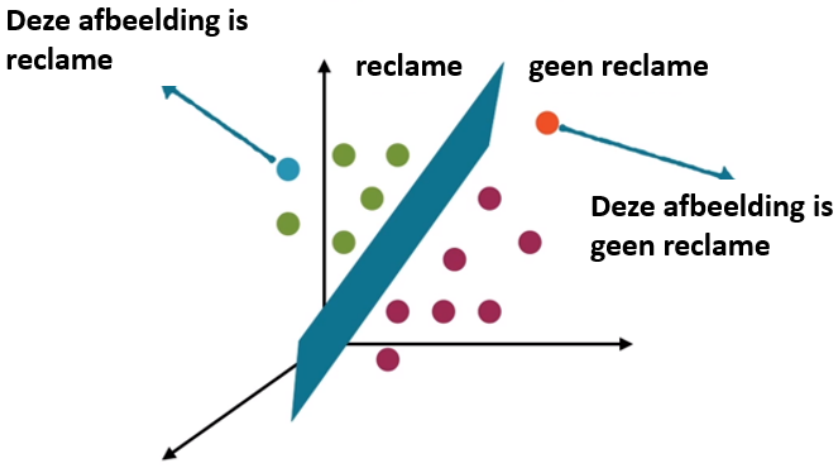


Hoe werkt het Support Vector Machine algoritme?

SVM classifiers zoeken dat vlak dat de datapunten het beste van mekaar scheidt. In ons geval gaat het SVM algoritme op basis van de punten van de training data een **grensvlak** zoeken tussen de 2 groepen afbeeldingen.



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.

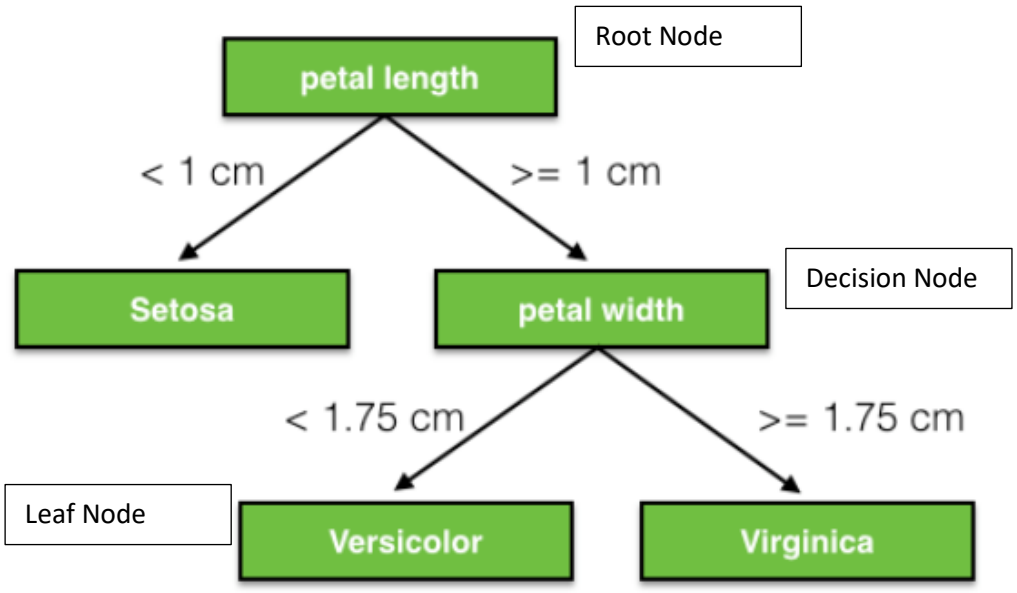


SVM kunnen we **enkel** gebruiken voor **binaire** classificatie (dus in 2 categorieën), dit in tegenstelling tot Naive Bayes waar we kunnen indelen in om het even welk aantal categorieën.

5.1.3 Decision Tree algoritme

Een Decision Tree is een 'boom' waarin elke knoop een kenmerk (feature) vertegenwoordigt, elke pijl een beslissing inhoudt en ieder blad één van de uitkomsten (categorisch) is. Een decision tree kan gebruikt worden voor multiple classification en dit algoritme werkt voor zowel categorische als continue afhankelijke variabelen.

Het knooppunt waar de boom begint staat bekend als de 'Root Node' en de andere knooppunten zijn 'Decision Nodes/Internal Nodes'. De uitkomst van de Decision Nodes zijn de 'Leaf-/Terminal Nodes'.

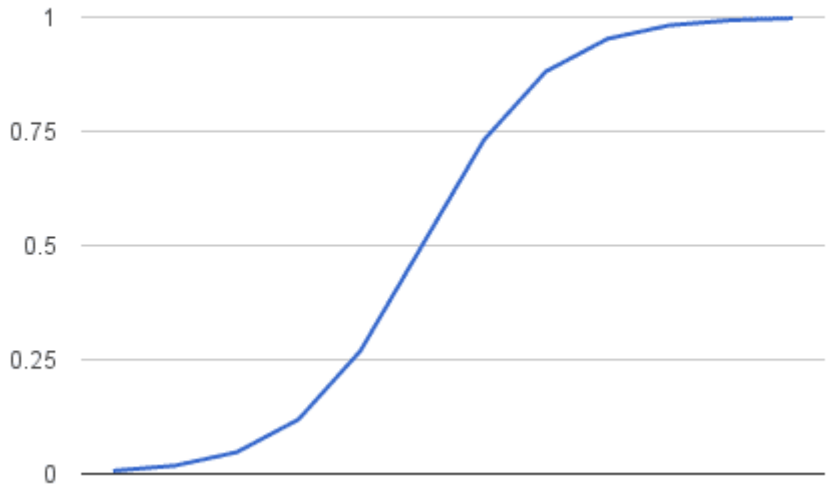


We passen dit Decision Tree algoritme toe op de Iris Data set om bloemen te classificeren in Setosa, Versicolor of Virginica.

Jupyter notebook: Machine_Learning_1_Classification_Iris

5.1.4 Logistic Regression Algoritme

Deze methode leent zijn naam aan de functie die aan de basis ligt van dit algoritme: de logistic functie : $f(x)$: $\frac{1}{1+e^x}$ waarbij elke inputwaarde x omgezet wordt in een waarde tussen 0 en 1. Deze logistic functie heeft een typische S-vorm.



Wanneer we onze outcome y (binary classification) willen voorspellen op basis van één input variabele x, wordt gebruik gemaakt van onderstaande functie gecombineerd met formules uit de kansrekening.

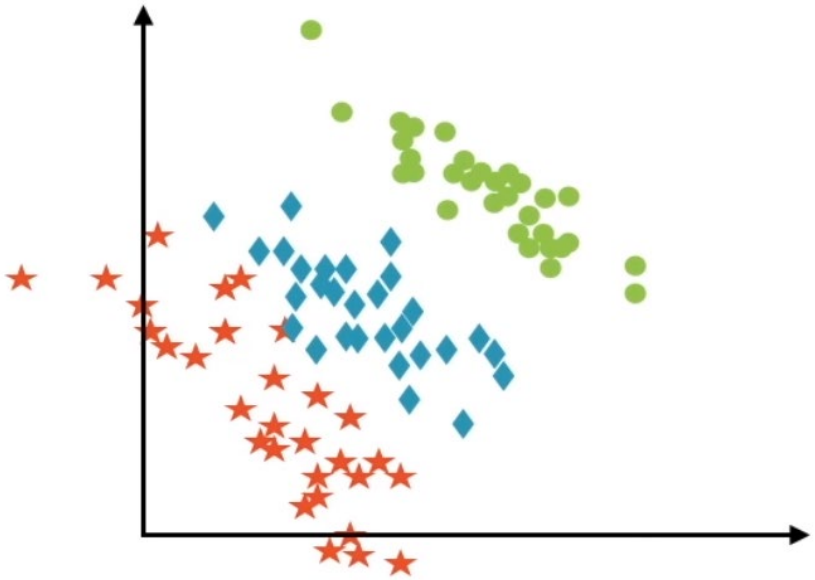
$$y = \frac{e^{b_0+b_1x}}{1 + e^{b_0+b_1x}}$$

Op basis van training data worden die b_0 en die b_1 bepaald die de beste predictie maken.

We passen dit Logistic Regression algoritme toe op de Titanic dataset om passagiers te classificeren als overlevende of niet.
Jupyter notebook: Machine_Learning_2_Classification_Titanic

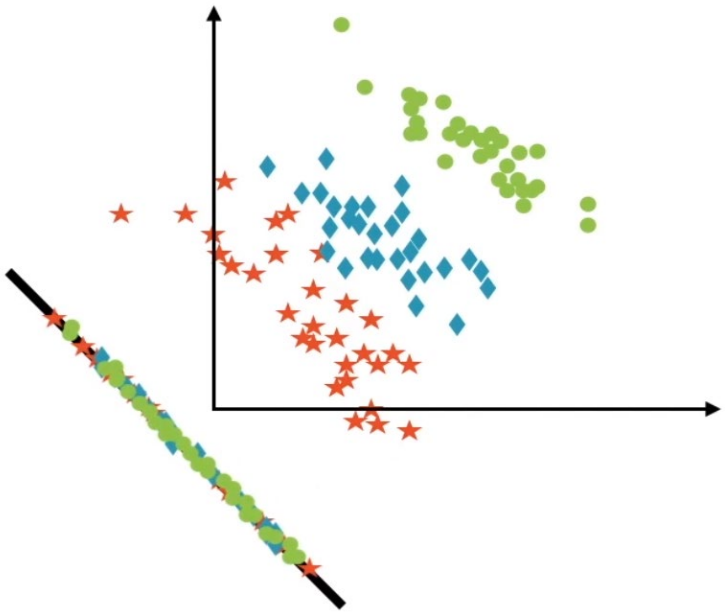
5.1.5 Linear Discriminant Analysis algoritme

Stel we hebben observaties die ingedeeld zijn in 3 groepen (groen, blauw, rood).

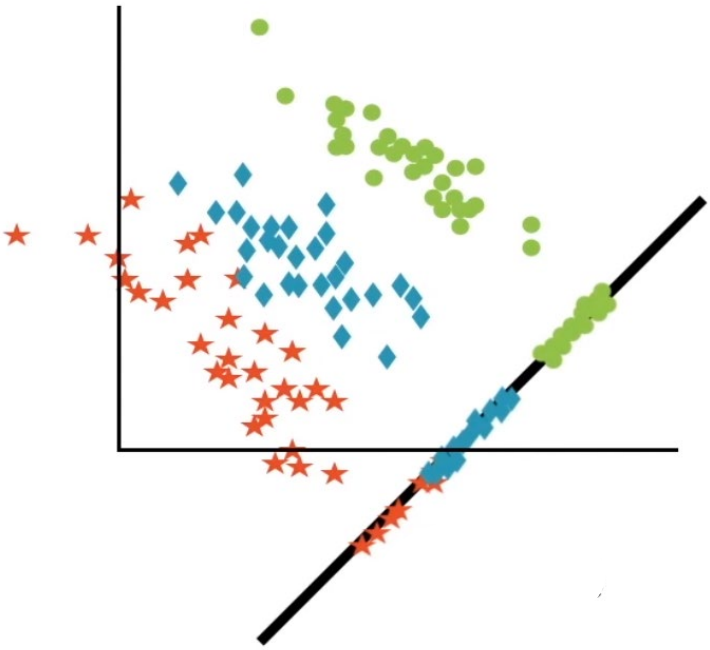


Het Linear Discriminant algoritme gaat nu op zoek naar deze as waarbij de afstanden tussen de projecties van die punten op deze as maximaal is.

Slechte keuze...



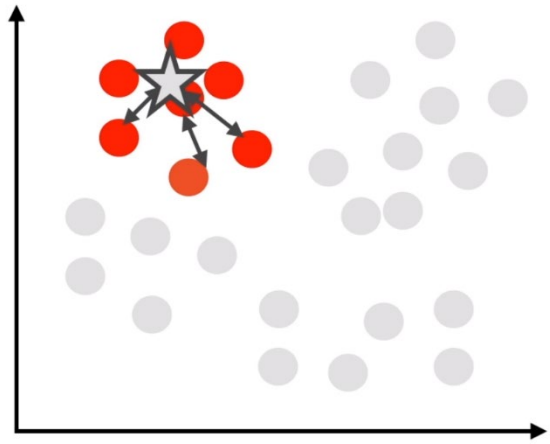
Beste keuze...



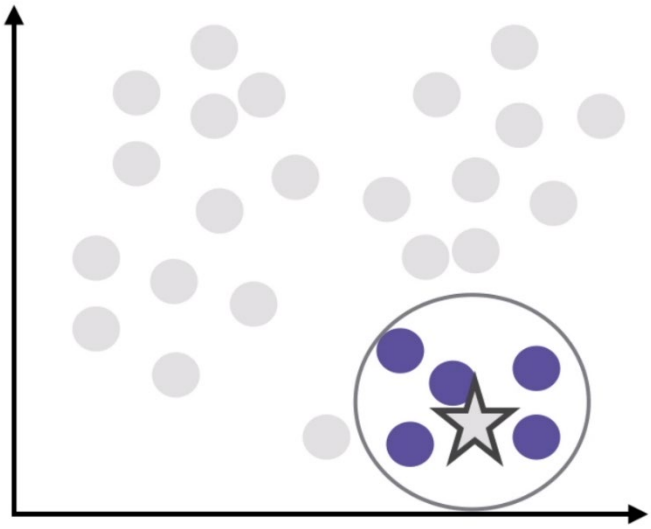
5.1.6 Nearest Neighbor algoritme

Op basis van data die gelabeld (supervised) is, zoeken we voor het nieuwe object de kortste buren. Buren worden bepaald op basis van afstand.

K - nearest neighbor classificatie



Radius nearest neighbor classificatie



5.2 Supervised ML: Regressie

Met behulp van regressie trachten we een continue variabele (= output) te voorspellen. Hier verschilt regressie van classificatie waar de output categorisch is. Verder weten we dat deze output afhangt van / beïnvloed wordt door sommige andere variabelen (input).

Dit is wat regressie doet: regressie zoekt de functie die het verband tussen de afhankelijke variabele en onafhankelijke variabelen zo goed mogelijk benadert. Is dit verband lineair, kwadratisch, exponentieel, … ?

Lineaire regressie is veruit de meest gebruikte en gekende regressie techniek.

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 (dus supervised Machine Learning).

Theorie achter eenvoudige lineaire regressie

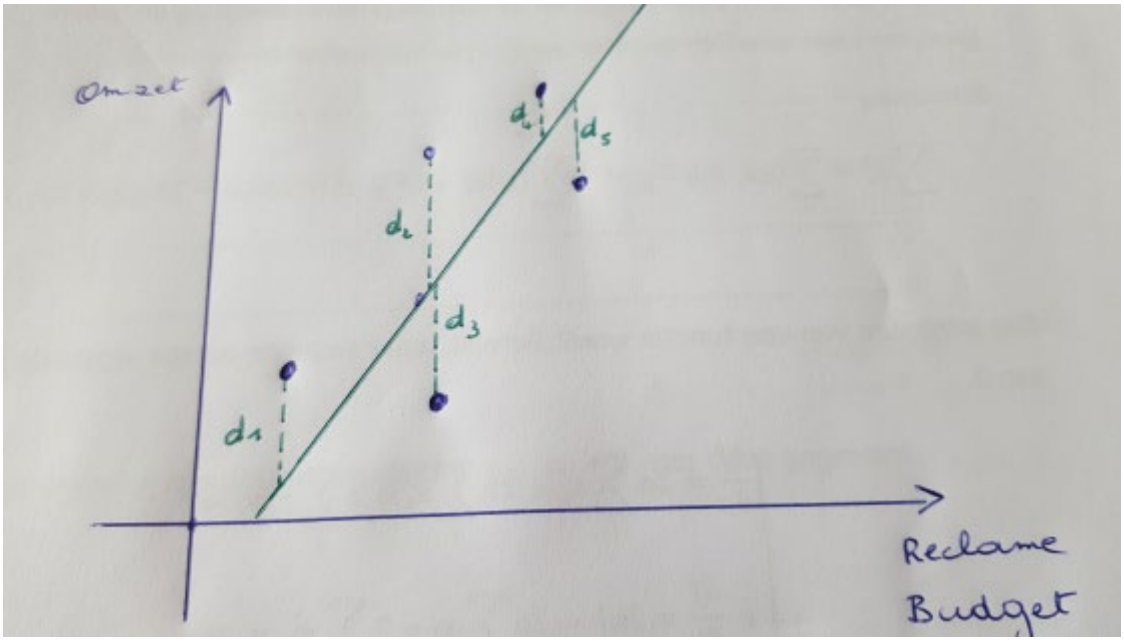
Een belangrijk aspect bij het op de markt brengen van een product is de hoeveelheid geld die aan reclame wordt besteed. Doel is natuurlijk dat reclame maken ook de verkoop van je product stimuleert.

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 bepalen aan de hand van data; 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: $a = \frac{\sum x_i y_i - n\bar{x}\bar{y}}{\sum x_i^2 - n\bar{x}^2}$ en $b = \bar{y} - a\bar{x}$

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.

$$\bar{x} = 3 \quad \bar{y} = 2 \quad n = 5 \quad \sum_{i=0}^{5} x_i y_i = 1 * 1 + 2 * 1 + 3 * 2 + 4 * 2 + 5 * 4 = 37$$

$$a = \frac{37 - 5*3*2}{55 - 5*3^2} = 0.7 \quad b = 2 - 0.7 * 3 = -0.1 \quad \rightarrow \quad y = 0.7 * x - 0.1$$

We passen dit Linear Regression algoritme toe op de Boston Houses dataset om de huisprijs te voorspellen.

Jupyter notebook: Machine_Learning_3_Regression_Boston

Enkele andere voorbeelden van regressie:

- Wat zal het rendement zijn van een aandeel op een bepaalde datum? Dit rendement kan afhangen van meerdere variabelen (= onafhankelijke variabelen) bvb dag van de week, dag van de maand, het rendement van de afgelopen dagen, andere producten op de markt, ….
- Als de wachttijd vergroot, welk effect heeft dit op klanttevredenheid?
- Wat is de verwachte verkoop van een bepaald product in een bepaalde week?
- 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, …
- gezichtsherkenning: Wanneer je naar een foto van een gezicht kijkt, 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.

5.3 Unsupervised ML: Clustering

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.

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 t.o.v. elkaar (dus de inter cluster gelijkenissen worden geminimaliseerd).

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

Voorbeeld: je hebt een grote groep mensen die actief zijn op social media. Op basis van gemeenschappelijke kenmerken deelt het clusteringalgoritme deze gebruikers op in clusters (groepen) zodanig dat in één cluster de gebruikers “vergelijkbaar” zijn met elkaar. 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, …). Wanneer een nieuwe gebruiker zich aanmeldt op social media, dan kan je classificatie gebruiken om deze user toe te kennen tot één of andere groep die ontstaan is door clustering.

Werkwijze:

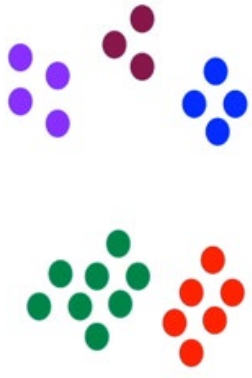
Van alle mensen (los van clusters) worden features verzameld: leeftijd, woonplaats, frequentie in het gebruik van bepaalde topics, keywoorden die vaak gebruikt worden op social media, …

Op deze manier wordt aan elke persoon een N – tupel van cijfers gekoppeld.

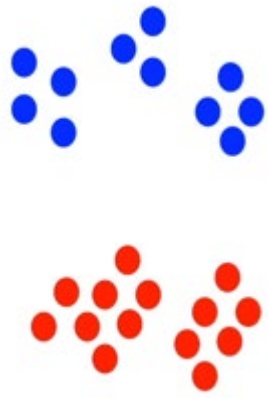
Dat N - tupel stelt een punt in de N – dimensionale ruimte voor.

Het clustering algoritme gaat groepen van mensen 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 …

Bestaande clustering Algoritmes:

K – means clustering: dit werken we iets dieper uit op volgende pagina

K – median clustering

Hiërarchical clustering

Density-based clustering

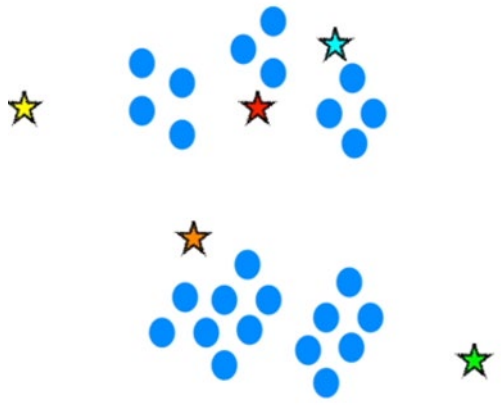
Distribution-based clustering

K – Means algoritme voor het clusteren van documenten

Gegeven een bepaalde set van documenten (artikels, reviews, tweets, boeken, webpagina's, …) , groepeer (cluster) deze op basis van gelijkenis. Na de clustering ontdek je misschien dat er interessante thema's zijn binnen de clusters: zelfde auteur, zelfde uitgiftejaar, 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. Cfr. "term frequency representation". Elk document wordt op deze manier een N - tupel van cijfers. Dit punt in de N – dimensionale ruimte wordt uitgezet in een N - dimensionaal assenstelsel.
3. Het K – means algoritme deelt onze data in K groepen in waarbij K op voorhand bepaald wordt door de gebruiker.

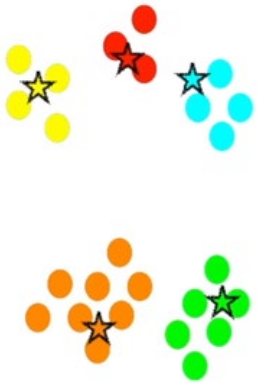
STAP 1: We beginnen met het initialiseren van een set van middens (K means = K middens van de clusters die we willen vinden). Het aantal clusters wordt vastgelegd door de gebruiker (zeg 5). De initiële middens kunnen random zijn maar er zijn technieken om deze initiële waarden op een zodanige manier 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 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.



5.4 Recommendations

Dit soort van ML probleem houdt zich bezig met “**aanbevelingen**” doen, de klant nieuwe noden laten ontdekken. Op basis van eerder gedrag van **jezelf** (opzoeken op internet, luisteren naar bepaalde liedjes, aankopen in een grootwarenhuis, …) en/of **vergelijkbare gebruikers** worden items waar de gebruiker mogelijk nog in geïnteresseerd is, voorspeld zelfs nog voor hij/zij het zelf al beseft. Gepersonaliseerde “aanbevelingen” winnen aan belang. Ze hebben een (positieve) invloed op o.a. koopgedrag. Het kunnen voorspellen van gebruikerstevredenheid is hier nauw mee verbonden. Dit is een erg grote business geworden, zeker in de online wereld.

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.

Voorbeeld:

- Bezoek eender welke “commerciële” website en je krijgt reclame over producten die je misschien wel leuk vindt en eventueel zal aankopen. Je zoekt online informatie op voor een nieuwe GSM ➔ je krijgt reclame over telefoonhoesjes…

- Wanneer je Netflix of Spotify of… gebruikt, krijg je aanbevelingen voor programma’s, films, liedjes die je misschien leuk vindt gebaseerd op hetgeen je reeds bekeken, geluisterd hebt.

- 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.

- 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.

Hoe gaat dit **praktisch** in zijn werk?

Hoe kunnen we aanbevelingen doen op basis van browser geschiedenis?

Gebruik het aantal keer 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.

Wanneer je een product koopt, krijg je vaak meldingen in de aard van: andere bezoekers kochten ook dit… Hoe worden deze meldingen geselecteerd?

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

Hoe vinden we de top 10 films die interessant zijn voor een gebruiker?

We beschikken over een dataset met movie ratings en verder ook over het “movie profiel” van onze gebruiker.
We zoeken nu naar vergelijkbare users met onze gebruiker: users die dezelfde films wel / niet appreciëren.
Op basis van deze “neighbor” users wordt de top 10 meest interessante films voor onze gebruiker voorgesteld.

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.

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

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.

Jupyter Notebook: Machine_Learning_4_Recommendations_Movie

6. 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)
- Building Machine Learning models in Python with scikit-learn (Janani Ravi)
- Preparing data for Machine Learning (Janani Ravi)
- Reducing Dimensions in Data with scikit-learn (Janani Ravi)

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/

Seminarie:

- Infofarm: An introduction to Machine Learning (Anaïs Ools, Glenn Cich en Lorenz Feyen)

  Notebooks: Machine_Learning_1_Classification_Iris
  Machine_Learning_3_Regression_Boston
  Machine_Learning_4_Recommendations_Movie