Unsupervised learning met K-means clustering

Voorbereiding bachelorproef

Arno Moonens
Richting: 3e Bachelor Computerwetenschappen
Email: arno.moonens@vub.ac.be
Rolnr.: 500513

Promotor: Yann-Michaël De Hauwere
Begeleiders: Maarten Devillé en Peter Vrancx

Inhoudsopgave

1 Abstract

Bij machine learning heeft men het doel om algoritmen te maken die kunnen leren en taken steeds beter kunnen uitvoeren. Bij supervised learning is het de bedoeling om een functie te maken die de output waarde voorspelt bij een bepaalde input waarde. Bij unsupervised learning is er zo geen output-waarde om na te kijken of de voorspelling correct is. Het is daarom de bedoeling om structuur te vinden in de data. Een van deze manieren is clustering. We zien dat deze manier data groepeert. Deze groepen data noemt men clusters. Hierbij is het de bedoeling dat data binnen clusters dicht bij elkaar staan en data van verschillende clusters ver van elkaar.

K-means is een van de algoritmen om clustering uit te voeren. We leggen uit dat de *MacQueen* implementatie een is die itereert en vrij simpel werkt. Het doet echter wel wat werk te veel. Daarom introduceren we het *filtering* algoritme dat door gebruik van een kd-boom dit probeert op te lossen. We bekijken de performantie van beide implementaties.
Er blijven echter nog wat problemen bestaan. Zo bekijken we hoe we het juiste aantal clusters zoeken die het algoritme moet genereren en trachten we door *random initialization* het probleem op te lossen dat *K-means* niet altijd een globaal optimum bereikt.
Karakteristieken die echter niet simpel op te lossen zijn, zijn de bolvorm van de gegenereerde clusters en de invloed van veel data die zich dicht bij elkaar bevinden.

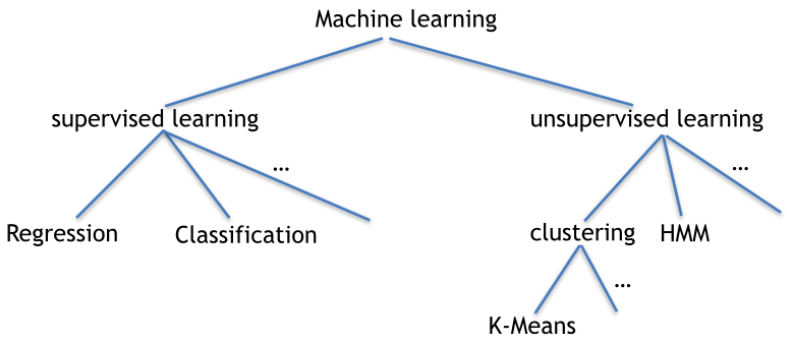
2 Introductie

Tegenwoordig wordt er een overvloed aan data verzameld, bijvoorbeeld wat je koopt of welke websites je bezoekt. De uitdaging bestaat er echter in om deze informatie te verwerken en er nuttige besluiten uit te kunnen trekken. Machine learning kan hier toe bijdragen en bevat vele technieken en algoritmen om ons te helpen de verkregen data te begrijpen.
Na het uitleggen van het doel van machine learning hebben we het over de 2 onderdelen ervan, namelijk supervised en unsupervised learning. We verdiepen ons in unsupervised learning en leggen clustering uit samen met het *K-means* algoritme. Hier hebben we het over 2 implementaties. De *MacQueen* implementatie en het *filtering* algoritme.
Naast het bespreken van de performantie van dit algoritme hebben we het ook over de karakteristieken van *K-Means* clustering.
Ten slotte wordt er uitgelegd wat de inhoud is van mijn bachelorproef en hoe het *K-means* algoritme hier voor nuttig kan zijn.

3 Probleemafbakening en overzicht

3.1 Machine learning

Bij machine learning is het de bedoeling om iets uit te voeren dat niet expliciet geprogrammeerd is door de gebruiker. Men kan spreken over een bepaalde taak die men moet uit voeren. Alvorens het uitvoeren heeft men een hoeveelheid ervaring met het uitvoeren van deze taak en een bepaalde kans tot het succesvol uitvoeren. De bedoeling is dat men na het uitvoeren van de taak meer ervaring heeft en de kans tot succes dus ook groter wordt.
Een voorbeeld van het gebruik van machine learning is *Optical Character Recognition* [8]. Hierbij wordt tekst herkend in een afbeelding. Door manueel te tonen welk teken overeenkomt met wat voor lijnen, kan een programma leren hoe een bepaald teken eruit ziet. Hoe meer deze taak uitgevoerd wordt, hoe groter de ervaring en dus de succeskans om tekst in een afbeelding te herkennen.
In figuur 1 is een subset van velden en algoritmen van machine learning te zien.



Figuur 1: Onderverdeling van verschillende gebieden binnen machine learning

3.1.1 Definities

- *Training voorbeeld*: De waarden van een aantal *features* samen met eventueel het bijhorende label (de *output* waarde).
- *Training set*: Een verzameling van training voorbeelden.
- *Feature*: Een eigenschap oftewel een input waarde van een voorbeeld. Dit is bijvoorbeeld de grootte van een huis, het aantal slaapkamers, ...

3.2 Supervised learning

Hierbij gaat het programma leren van reeds gelabelde training voorbeelden om zo het label te voorspellen van ongelabelde voorbeelden. Men zoekt dus naar een functie die voor de features (inputs) een label (output) teruggeeft.
Deze manier van leren kan op zijn beurt nog onderverdeeld worden in *regression* en *classification*.
Bij *regression* zijn de labels, de output waarden die men dus wilt voorspellen, numeriek en continu van aard. Men kan bijvoorbeeld de prijs van een huis proberen te voorspellen afhankelijk van de grootte en eventueel nog andere features [1].
Bij *classification* zijn er maar een beperkt aantal waarden voor een label van een training voorbeeld. Dit kunnen numerieke waarden zijn, maar dit is niet verplicht. Zoals de naam zelf zegt, is het bij *classification* de bedoeling om te voorspellen tot welke klasse een training voorbeeld behoort. Een voorbeeld hiervan is het proberen bepalen of een ontvangen email al dan niet spam is [1].

3.3 Unsupervised learning

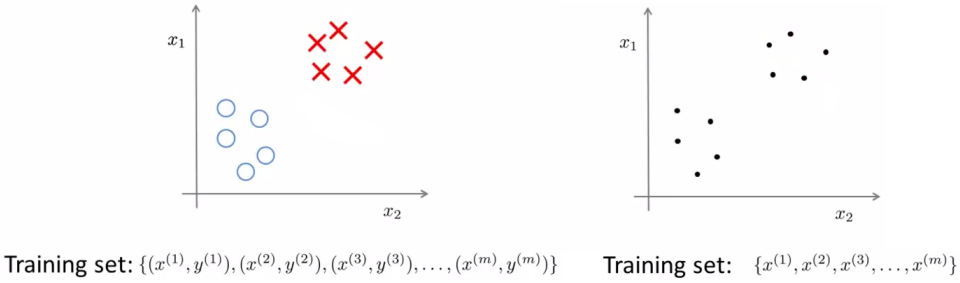
Bij unsupervised learning heeft geen enkel training voorbeeld een label. Het is hier dan ook niet de bedoeling om afzonderlijk labels voor training voorbeelden te gaan voorspellen omdat er simpelweg geen 'juiste' of 'foute' voorspellingen zijn (het al dan niet correct toekennen van een label aan een training voorbeeld) [9].
Men gaat hier echter op een ander manier naar structuur zoeken in de training set. Deze gevonden representaties kunnen dan gebruikt worden voor onder andere het detecteren van training voorbeelden met waarden die veraf liggen van die van andere voorbeelden of als hulpmiddel voor andere algoritmen.
Er zijn verschillende benaderingen bij unsupervised learning:

- *Blind signal separation*: Hierbij ontvangt men een vermengd signaal en gaat men op zoek naar de verschillende bronsignalen die het gemengd signaal vormen. Dit kan bijvoorbeeld gebruikt worden om de stem van een bepaald persoon te onderscheiden wanneer verschillende personen tegelijk aan het praten zijn. [10] [11]

- *Hidden Markov model*: Maakt gebruik van een staat-transitie model [12]. Het wordt vooral gebruikt om patronen te zoeken in een tijdreeks. Zo kan men bijvoorbeeld bij een persoon een bepaalde aandoening detecteren aan de hand van de hartslag [13].

- *Clustering*: Dit wordt in het volgende hoofdstuk beschreven.

In figuur 2 is het verschil tussen een training set bij supervised en supervised learning te zien.



Figuur 2: Links is een training set bij supervised learning te zien. Er zijn hier 2 mogelijke output-waarden. Rechts ziet men een training set bij unsupervised learning. Men moet hier een bepaalde structuur vinden in de training set.
Bron: https://class.coursera.org/ml-005

3.4 Clustering

Clustering is een vorm van unsupervised learning waarbij men gelijksoortige training voorbeelden gaat groeperen. Zo een groepen noemt men clusters.
Het is de bedoeling dat training voorbeelden binnen een cluster sterk op elkaar gelijken en de gelijkenis met training voorbeelden uit een andere cluster eerder klein is [15]. De 'gelijkenis' wordt bepaald door de hoeveelheid dat de waarden van features van verschillende training voorbeelden van elkaar verschillen (een formele definitie volgt later). Voorbeelden en toepassingen van clustering zijn:

- Het bepalen van T-shirt groottes zodanig dat er weinig verschillende maten zijn maar waarvan elke maat toch voor een grote groep past.
- Beeldherkenning: groeperen van punten in een afbeelding (pixels) die tot hetzelfde object behoren.
- Het construeren van plant- en dier taxonomiën.
- Galaxie formaties bestuderen.
- Het efficiënt plaatsen van (computers in) datacenters.

Voor het opstellen van zogenaamde clusters bestaan er verschillende soorten algoritmen.
Bij *hierarchical clustering* krijgt elk training voorbeeld initieel zijn eigen cluster en gaat men daarna dichtbijzijnde clusters samenvoegen. Op deze manier ontstaat er een boomstructuur [14].
Het gebruik van *partitional clustering* daarentegen resulteert niet in een hiërarchie, maar slechts in een lijst van clusters. Als je meegeeft dat je een bepaald aantal clusters wilt, worden al deze clusters tegelijk gevonden.
Uiteraard kan men bij *hierarchical clustering* ook een lijst van clusters bekomen, door de clusters op een bepaald niveau van de gegenereerde boom te aanschouwen.

Een van deze *partitional clustering* algoritmen is het *K-Means* algoritme.

3.5 K-Means

Voor *K-Means* zelf bestaan er verschillende implementaties. We bekijken eerst de implementatie van MacQueen.

3.5.1 MacQueen implementatie

Deze implementatie werkt in 4 stappen. De algemene werking wordt hier uitgelegd [16].
Allereerst initialiseert men K centroids op een willekeurige "plaats". Dit betekent dus dat men zelf een artificieel training voorbeeld aanmaakt met evenveel features als die bij de reeds aanwezige training voorbeelden en waarbij de waarden voor deze features willekeurig gekozen zijn.
Vervolgens worden de volgende stappen steeds herhaald tot er aan een bepaalde stopconditie voldaan wordt:

- Men gaat elk training voorbeeld toewijzen aan de centroid die het dichtste bij is. Dit betekent dus dat het verschil van waarden van de features kleiner is dan die tussen een andere centroid. Je hebt nu voor elke centroid een groep van training voorbeelden die tot deze centroid behoren. Deze groep is een cluster.

- Je berekent voor elke cluster het gemiddelde van zijn training voorbeelden. Je gaat dus per feature berekenen wat het gemiddelde is van de waarde van deze feature bij alle training voorbeelden.

- Je gaat de centroid van elke cluster verplaatsen naar het zojuist berekende gemiddelde van deze cluster. De waarden van de features van de centroid worden dus gewijzigd naar de zojuist berekende gemiddelden van de features.

In figuur 3 zijn verschillende iteraties van dit algoritme te zien.
Om dit algoritme werkelijk te implementeren moeten er uiteraard een aantal zaken concreet ingevuld worden:

- Hoe vindt men de dichtstbijzijnde centroid?
- Wanneer mag het algoritme stoppen.

We definiëren eerst wiskundig de training set en centroids.

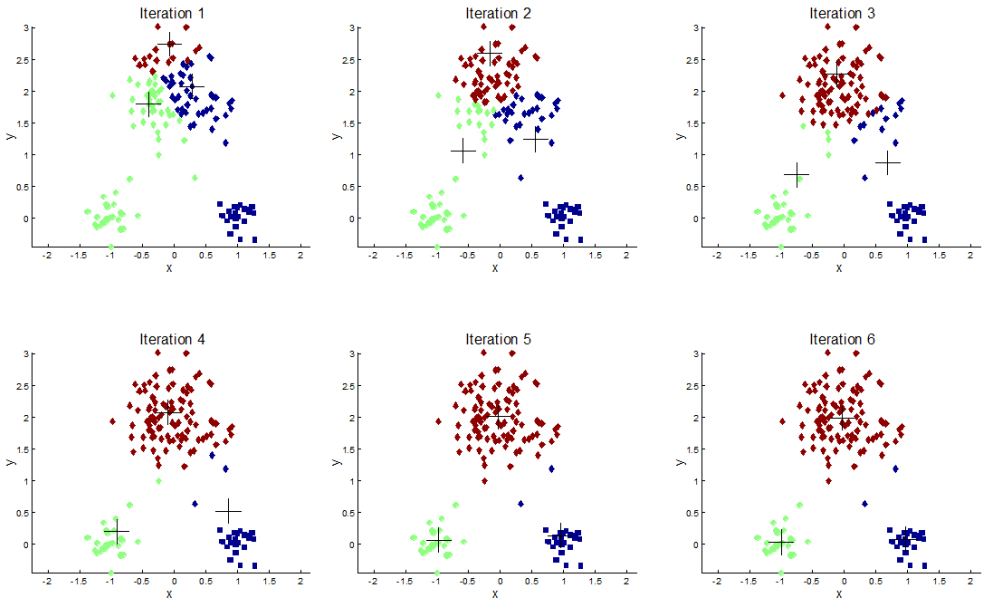
n	aantal features.	(1)
$x^{(i)}$	feature vector van het i'de training voorbeeld.	(2)
$x_j^{(i)}$	waarde van de j'de feature bij het i'de training voorbeeld.	(3)
$c^{(i)}$	Index van een cluster waar het i'de training voorbeeld aan toegewezen is.	(4)
μ_k	De k'de cluster centroid.	(5)

De dichtstbijzijnde centroid wordt voor dan elk training voorbeeld als volgt bepaald:

$$c^{(i)} = \underset{k}{\operatorname{argmin}} \|x^{(i)} - \mu_k\|^2 \quad [1]$$

Dit betekent dat men de centroid zoekt waarbij de euclidische afstand het kleinst is.
Men kan het algoritme laten stoppen bij het voldoen van een van deze condities:

- Er zijn een bepaald aantal iteraties voltooid. Dit is moeilijk op voorhand vast te stellen omdat de samenstelling (welke training voorbeelden tot welke centroid en dus cluster behoren) van clusters mogelijk nog regelmatig kan veranderen.
- De samenstelling van clusters verandert niet meer of minder dan een bepaalde waarde.
- De samenstelling is optimaal genoeg (wat dit betekent wordt uitgelegd in 3.5.4).

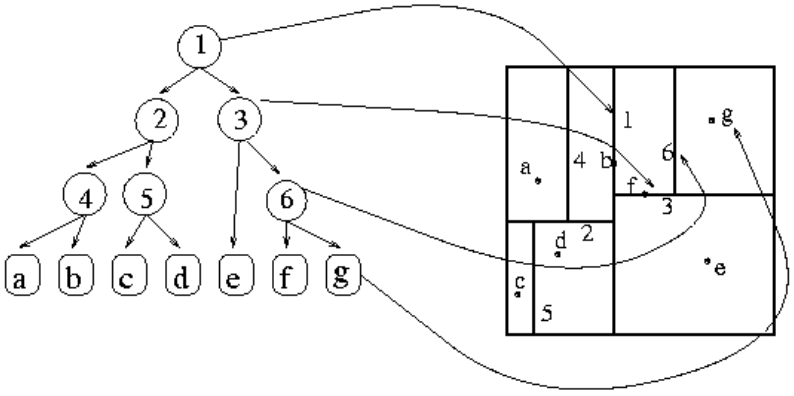


Figuur 3: Voorbeeld van 6 iteraties van het *K-means* algoritme bij 2-dimensionele data. Bij elke iteratie worden de centroids verplaatst naar het midden van de toegewezen training voorbeelden en worden training voorbeelden hertoegewezen aan hun dichtstbijzijnde centroid.
Bron: `https://apandre.files.wordpress.com/2011/08/kmeansclustering.jpg`

3.5.2 Filtering algoritme

Dit is slechts een variatie op de *MacQueen* implementatie. Het maakt gebruik van een specifieke datastructuur, namelijk een kd-boom [5].
Het *kd*-gedeelte staat hier voor k-dimensionaal. Elke dimensie stelt een feature van een training voorbeeld voor.
Men begint bij een balk (ook wel *hyperrectangle* genoemd) die alle training voorbeelden bevat. Deze balk kan opgesplitst worden orthogonaal op zijn langste zijde. Deze 2 delen zijn dan nodes, oftewel cellen genoemd, van de originele balk. In elke dimensie en dus voor elke feature heeft een cel dan een minimum- en maximumwaarde.
De bekomen cellen kan men opnieuw elk in 2 delen. Op deze manier ontstaat er een binaire boom. Hoe verder men gaat in deze boom, hoe kleiner de cellen en dus ook hoe minder training voorbeelden een cel bevat. Een cel met minder dan een bepaald aantal trainingvoorbeelden (bijvoorbeeld slechts 1) wordt een blad genoemd.
Het opsplitsen zelf in een bepaalde dimensie kan gebeuren in het midden van een cel, dus in het midden van de minimum- en maximumwaarde van

de feature in kwestie.
Een andere manier is om te splitsen op de mediaan in die dimensie van de training voorbeelden.
Een voorbeeld van een kd-boom is te zien in figuur 4.

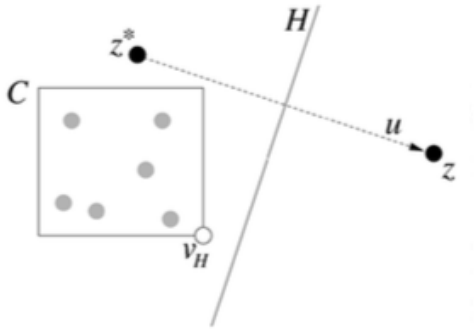


Figuur 4: Een voorbeeld van een opgesplitst vlak en de bijhorende kd-boom in 2 dimensies. Men heeft er hier voor gekozen om te splitsen op de mediaan van de training voorbeelden.
Bron: `http://upload.wikimedia.org/wikipedia/commons/b/b6/3dtree.png`

Het algoritme begint met het genereren van deze kd-boom. Voor elke node u houden we ook het aantal training voorbeelden ($u.count$) in zijn cel bij en een artificieel training voorbeeld dat de vector som is van de features van alle training voorbeelden in de cel ($u.sum$).
Daarbuiten heeft elke node ook een aantal kandidaat-centroids. Dit zijn centroids die voor een training voorbeeld in de cel de dichtstbijzijnde centroid is. Voor de wortel van de kd-boom zijn alle centroids kandidaten-centroids.
Voor elke node zoekt men de centroid z^* die zich het dichtst bij het midden van de cel bevindt (dit midden is $u.sum/u.count$). Vervolgens zoekt men of er andere centroids zijn die voor een bepaald training voorbeeld in de cel dichterbij zijn dan z^*. Elke centroid die die eigenschap niet heeft wordt "weggefilterd".
Als er uiteindelijk geen centroids meer overblijven is z^* de enige kandidaat. We kunnen dan alle training voorbeelden in deze cel toewijzen aan deze centroid, zodat er reeds een (onvolledige) cluster ontstaat.
Als er wel alternatieve centroids overblijven, moet men het zonet uitgelegde proces opnieuw uitvoeren voor de kinderen van de node.
Als de node een blad was moet elk training voorbeeld apart aan de dichtst bijzijnde centroid toegewezen worden.

In figuur 5 is te zien dat z nog een andere centroid is en dat men dus moet kijken naar de kinderen van de node als een bepaald training voorbeeld zich rechts van een lijn bevindt die orthogonaal staat op het midden van de lijn $\overline{z^*z}$. Het training voorbeeld waarmee men dit nakijkt is diegene die zich het dichtst bij z bevindt.

Deze implementatie is een verbetering ten opzichte van de *MacQueen* implementatie omdat men niet telkens in elke iteratie voor elk training voorbeeld de dichtstbijzijnde centroid moet zoeken. Dit is slechts nodig als men bij een blad in de kd-boom terecht komt.
De kd-boom moet ook niet elke iteratie opnieuw gegenereerd worden omdat deze gebaseerd is op de training voorbeelden. In tegenstelling tot de centroids, veranderen training voorbeelden niet elke iteratie van plaats.



Figuur 5: Representatie in 2 dimensies waarbij beslist wordt of z nog een mogelijke kandidaat-centroid is.
Bron: `http://www.cs.umd.edu/~mount/Papers/pami02.pdf`

3.5.3 Performantie

We hebben het hier over de performantie van zowel de *MacQueen* implementatie als het *filtering* algoritme.
Bij *MacQueen* gaat men t iteraties voor elk training voorbeeld de afstand berekenen tot alle centroids. De performantie van deze implementatie is daarom $O(tkN)$, met k het aantal centroids en N het aantal training voorbeelden [1].
De performantie van het *filtering* algoritme is:

$$O(k(\frac{c\sqrt{d}}{\epsilon})^d + 2^d k \log n + \frac{dn}{\rho^2(1-\epsilon)^2}) \quad \text{(Zie [5] voor het bewijs)}$$

De variabelen in deze formule betekenen het volgende:

k Aantal te bekomen clusters. (6)

d Aantal dimensies (i.e. aantal features) van training voorbeelden. (7)

n Aantal trainingvoorbeelden. (8)

c Constante. (9)

ϵ $0 < \epsilon < 1 - \delta$ met δ de afstand tussen een centroid en het midden van zijn cluster. (10)

ρ Hoever clusters van elkaar gescheiden zijn. (11)

3.5.4 Optimalisatie doel

Opdat een samenstelling zo optimaal mogelijk is moet het gemiddelde van de afstanden van een training voorbeeld tot zijn toegewezen centroid zo klein mogelijk zijn. Dit gemiddelde wordt berekend aan de hand van volgende formule:

$$J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K) \frac{1}{m} \sum_{i=1}^{m} \|x^{(i)} - \mu_{c^{(i)}}\|^2 \quad [1]$$

Met m het aantal training voorbeelden en K het aantal centroids en dus clusters.
Dit wordt ook wel de kostfunctie genoemd. Men kan ervoor kiezen om het algoritme te laten stoppen als deze kost kleiner is dan een bepaalde waarde

Zoals eerder gezegd, worden de centroids op willekeurige plaatsen geinitialiseerd in het begin van het algoritme. Dit kan er echter toe leiden dat het lang duurt eer het optimum is bereikt of het is zelfs mogelijk dat het algoritme stopt bij een lokaal optimum in plaats van een globaal optimum. Dit betekent dus dat het algoritme gestopt is met itereren, ook al was er nog duidelijk een betere toekenning van clusters. Een voorbeeld van een situatie met een lokaal optimum maar geen globaal is te zien in figuur 6. Om dit op te lossen maakt men gebruik van *random initialization*. Dit houdt in dat men het *K-means* algoritme meerdere keren gaat uitvoeren met telkens een willekeurige initialisatie van centroids.
Voor elk resultaat gaat men dan telkens de kost berekenen met de reeds geziene kostfunctie. Men kiest tenslotte het resultaat met de kleinste kost:

$$\underset{c^{(1)},\ldots,c^{(m)},\mu_1,\ldots,\mu_K}{\operatorname{argmin}} \quad J(c^{(1)}, \ldots, c^{(m)}, \mu_1, \ldots, \mu_K) \quad [1]$$

Figuur 6: Links worden de originele training voorbeelden getoond en rechts de training voorbeelden gekleurd afhankelijk van tot welke cluster ze behoren. Het is duidelijk dat er hier 3 groepen van training voorbeelden zijn. Bij het resultaat van het *K-means* algoritme zijn hier echter 2 clusters toegekend aan 1 groep data en 1 cluster aan 2 groepen data.
Bron: `http://image.slidesharecdn.com/k-meansclusteringpdf-140704050935-phpapp01/95/k-means-clustering-28-638.jpg?cb=1404811288`

3.5.5 Keuze van het aantal clusters

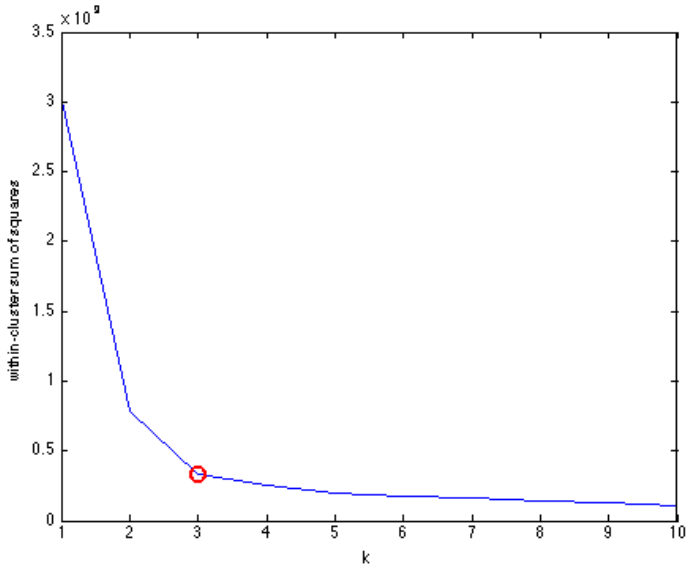
Voor het *K-Means* algoritme moet het aantal gewenste clusters, K, op voorhand meegegeven worden. Er worden dan evenveel centroids aangemaakt.
Voor het bepalen van dit aantal bestaat er echter geen formule die voor alle soorten data werkt. Het aantal clusters hangt af van de betekenis van de data en wat je met het clustering algoritme wil bereiken. Voor het kiezen K zijn er verschillende aanpakken mogelijk.

De eerste aanpak bestaat er in om het aantal clusters te kiezen in functie van het vooropgestelde doel. Je kan bijvoorbeeld kiezen om de markt voor T-shirts in 3 te delen (*small*, *medium* en *large*), en dan voor elke grootte de juiste afmetingen te kiezen. Hierbij gaat men dan voor 3 clusters kiezen ($K = 3$).

Men kan ook simpelweg de formule $K \approx \sqrt{\frac{n}{2}}$ gebruiken, met n het aantal training voorbeelden. Deze formule houdt echter geen enkele rekening met de aard van de data.

Een andere methode is de *elbow method* [1]. Men gaat hierbij een grafiek maken van de kost van het beste resultaat (gebruik makende van het reeds uitgelegde *random initialization*) in functie van het aantal clusters.

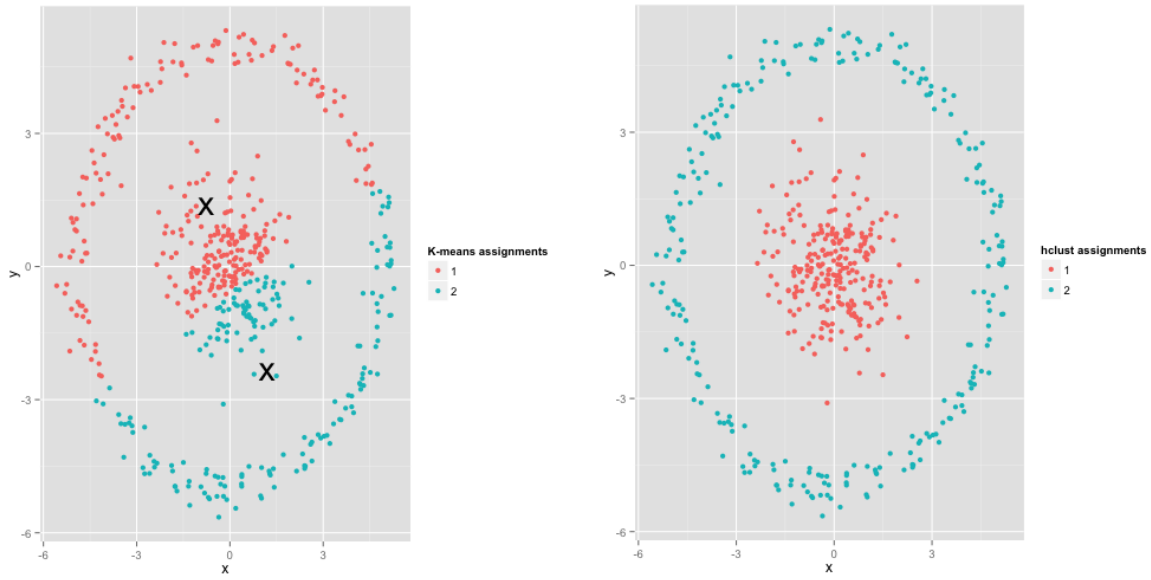
Bij het toepassen van deze methode op bepaalde data kan men dan mogelijk een 'knik' zien in de grafiek. Rechts van deze knik (dus bij het gebruik van meer clusters) daalt de kostfunctie dan niet zo veel meer als daarvoor. Het aantal clusters bijhorende bij dit knikpunt wordt dan gekozen bij het initialiseren van het algoritme. Een voorbeeld van deze grafiek is te zien in Figuur 7.



Figuur 7: Voorbeeld van een toepassing van de *elbow method*. Bij meer dan 3 clusters daalt de kost relatief weinig. 3 wordt hier gekozen als het aantal clusters ter initialisatie van het algoritme.
Bron: `https://cw.fel.cvut.cz/wiki/_media/courses/ae4b33rpz/labs/10_k-means/elbow_graph_hlt_letters.png`

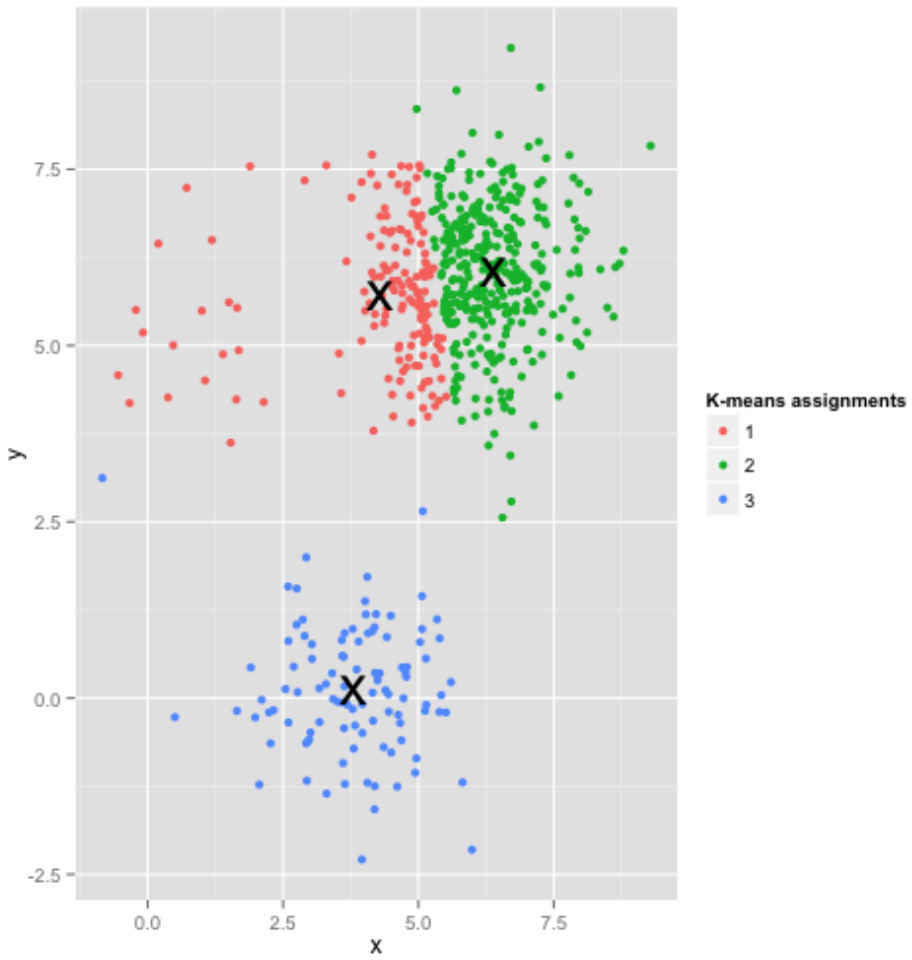
3.5.6 Problemen met cluster model

Zoals reeds gezegd worden bij *K-means* clustering training voorbeelden toegekend aan hun dichtst bijzijnde cluster. Het gemiddelde van de training voorbeelden en de centroid convergeren op termijn naar elkaar. Dit alles zorgt voor bolvormige cluster vormen of vormen waarbij zichtbare groepen data in stukken zijn "gesplitst" door een toekenning aan verschillende centroids [17]. Dit komt echter niet altijd overeen met de natuurlijke vorm van de groepen training voorbeelden. Een voorbeeld hiervan is te zien in figuur 8.



Figuur 8: Links is het resultaat van het *K-means* algoritme te zien en rechts het resultaat van een hiërarchisch cluster algoritme. Het is duidelijk dat er rechts een correcte toekenning van training voorbeelden is gebeurd. Door de manier waarop het *K-means* algoritme werkt komt het resultaat daarvan echter niet overeen met een logische toekenning en zijn aan 1 centroid training voorbeelden van 2 groepen toegekend.
Bron: http://varianceexplained.org/r/kmeans-free-lunch/

Een ander probleem heeft te maken met de hoeveelheid training voorbeelden die een groep bevat. Doordat een centroid steeds naar het gemiddelde van zijn training voorbeelden verplaatst wordt, heeft een groep met veel training voorbeelden veel meer invloed, ook wel gewicht genoemd, dan een groep met weinig training voorbeelden. De centroids gaan dan ook convergeren naar plaatsen waar er veel training voorbeelden zijn [18]. Een voorbeeld van dit probleem is te zien in figuur 9.



Figuur 9: In dit voorbeeld zijn 3 groepen training voorbeelden te zien. Omdat de groep rechts boven relatief veel training voorbeelden heeft zijn de centroids geneigd om daar naar te convergeren. Dit resulteert in 2 centroids die dicht bij elkaar liggen en een groep links boven met relatief weinig training voorbeelden die geen eigen cluster heeft.
Bron: http://varianceexplained.org/r/kmeans-free-lunch/

4 Conclusie

Bij machine learning gaat men proberen leren uit de data. We hebben gezien dat supervised en unsupervised learning 2 manieren zijn om te leren. Clustering hoort bij unsupervised learning en heeft als doel om clusters van data te vinden. De training voorbeelden in deze clusters

bevinden zich dicht bij elkaar.

We hebben *K-means* besproken als zijnde een algoritme om clusters te bekomen. 2 implementaties hiervoor werden uitgelegd, de *MacQueen* implementatie en het *filtering* algoritme. We bekeken ook hun performantie bekeken en besloten dat het *filtering algoritme* iets beter is omdat men niet voor elk training voorbeeld de dichtstbijzijnde centroid moest vinden.

Daarnaast bespraken we de "kost" van een resultaat van het algoritme en zagen we hoe *random initialization* een oplossing kan zijn voor het niet bereiken van een globaal optimum.
Een ander probleem is het bepalen hoeveel clusters we willen en dus hoeveel centroids we willen initialiseren. Dit is echter niet deterministisch te bepalen en hangt af van de soort data. Een goede manier is echter de *elbow method*.
Tenslotte zijn er problemen bij het *K-means* algoritme die niet meteen een oplossing voorhande hebben. Training voorbeelden worden steeds toegekend aan de centroid die zich het dichtste bij bevindt, maar dit zorgt echter niet altijd voor een correct en logisch resultaat. Daarbuiten wordt er geen rekening gehouden met "buitenstaanders" omdat een hoge concentratie van training voorbeelden op een bepaalde plaats meer invloed heeft op de locatie van centroids en dus clusters.

5 Beschrijving bachelorproef

Voor mijn bachelorproef is het de bedoeling om *tweets* (berichten op *Twitter*) te vinden die te maken hebben met bepaalde films. De tekst en metadata (locatie, tijd,...) kunnen dan gelinkt worden aan verdere info in verband met deze films op *web services* zoals *IMDB* of *Rotten Tomatoes*. Dit kan gaan over acteurs, genre, *rating*, aantal prijzen,...
Het project verloopt als volgt:

- *Tweets* worden opgezocht over films op basis van een hashtag of de naam van de film die in de tekst van de tweet staat. Deze *tweets* worden verzameld gebruik maken de van de Twitter API (*application programming*).
- De gevonden *tweets* worden opgeslagen in een database. Deze bevat naast de tekst zelf ook de metadata van de *tweets*.
- Op deze *tweets* wordt *sentiment analysis* uitgevoerd. Dit betekent dat men de tekst van de tweet gaat interpreteren en labels geven, zoals bijvoorbeeld positief/negatief, enthousiast/ingetogen,...

- Verdere info over de film op reeds genoemde *web services* worden opgezocht. Men zoekt vervolgens naar een link tussen info over de film en de populariteit met behulp van de *tweets*.
- Men zoekt ook naar een link tussen metadata van de *tweets* zelf. Dit gaat over links tussen het sentiment, de locatie en de tijd van publicatie van de tweet.
- De bevindingen worden gerapporteerd en gevisualiseerd.

Het *k-means* clustering algoritme kan op verschillende manieren bij het project van pas komen.
Men kan *tweets* met locatiegegevens uit de database halen en clustering toepassen op hun locatie. Op deze manier kan men te weten komen of er plaatsen zijn waar er veel over een bepaalde film getweet wordt en waar de film dus voor controverse zorgt.
Daarnaast kan men ook op het resultaat van de *sentiment analysis* clustering toepassen. Men kan bijvoorbeeld voor elke tweet, aan de hand van de tekst, berekenen hoe enthousiast men is en of men al dan niet blije gevoelens heeft. Dankzij clustering kan men dan zien of er bepaalde gevoelens over de film overheersen en hetzelfde zijn bij een groep *tweets*. Men zou bijvoorbeeld kunnen zeggen dat de ene cluster heel blij en enthousiast is, terwijl de andere ook zeer enthousiast is, maar heel negatieve gevoelens heeft over de film.

6 Referenties

6.1 Opgelegd door promotor/begeleiders

[1] Ng, A. "Machine Learning"
Opgehaald van `https://class.coursera.org/ml-005`

[2] Zaïane O.R. "Data Clustering"
Opgehaald van `http://webdocs.cs.ualberta.ca/~zaiane/courses/cmput695/F07/slides/ch6Clus-695-F07.pdf`

[3] Piech C. "K Means"
Opgehaald van `http://stanford.edu/~cpiech/cs221/handouts/kmeans.html`

[4] Coates, Adam, and Andrew Y. Ng. "Learning feature representations with k-means."

Neural Networks: Tricks of the Trade. Springer Berlin Heidelberg, 2012. 561-580.

[5] Kanungo, Tapas, et al. "An efficient k-means clustering algorithm: Analysis and implementation."
Pattern Analysis and Machine Intelligence, IEEE Transactions on 24.7 (2002): 881-892.

[6] Onbekend. "Clustering With K-Means in Python"
Opgehaald van `https://datasciencelab.wordpress.com/2013/12/12/clustering-with-k-means-in-python/`

[7] Onbekend. "K-means"
Opgehaald van `http://scikit-learn.org/stable/modules/clustering.html#k-means`

6.2 Extra

[8] Eikvil, Line. "OCR-optical character recognition." (1993).

[9] Ghahramani, Zoubin. "Unsupervised learning."
Advanced Lectures on Machine Learning. Springer Berlin Heidelberg, 2004. 72-112.

[10] Clifford GD. "Biomedical Signal and Image Processing", 2008

[11] Li Y. and Powers D. "BLIND SIGNAL SEPARATION/DECONVOLUTION." USING RECURRENT NEURAL NETWORKS, 2001

[12] Warakagoda N. "Definition of Hidden Markov Model."
`http://jedlik.phy.bme.hu/~gerjanos/HMM/node4.html`

[13] Wang, Peng, Haixun Wang, and Wei Wang. "Finding semantics in time series."
Proceedings of the 2011 ACM SIGMOD International Conference on Management of data. ACM, 2011.

[14] Steinbach, Michael, George Karypis, and Vipin Kumar. "A comparison of document clustering techniques."
KDD workshop on text mining. Vol. 400. No. 1. 2000.

[15] Boroš, Petr. "Clustering Analysis in Educational Data."
Diss. Bachelors thesis. Masarykova univerzita, Fakulta informatiky, 2013 (to appear).

[16] Wagstaff, Kiri, et al. "Constrained k-means clustering with background knowledge."
ICML. Vol. 1. 2001.

[17] Jain, Anil K. "Data clustering: 50 years beyond K-means."
Pattern recognition letters 31.8 (2010): 651-666.

[18] Robinson D. "K-means clustering is not a free lunch"
Opgehaald van `http://varianceexplained.org/r/kmeans-free-lunch/`