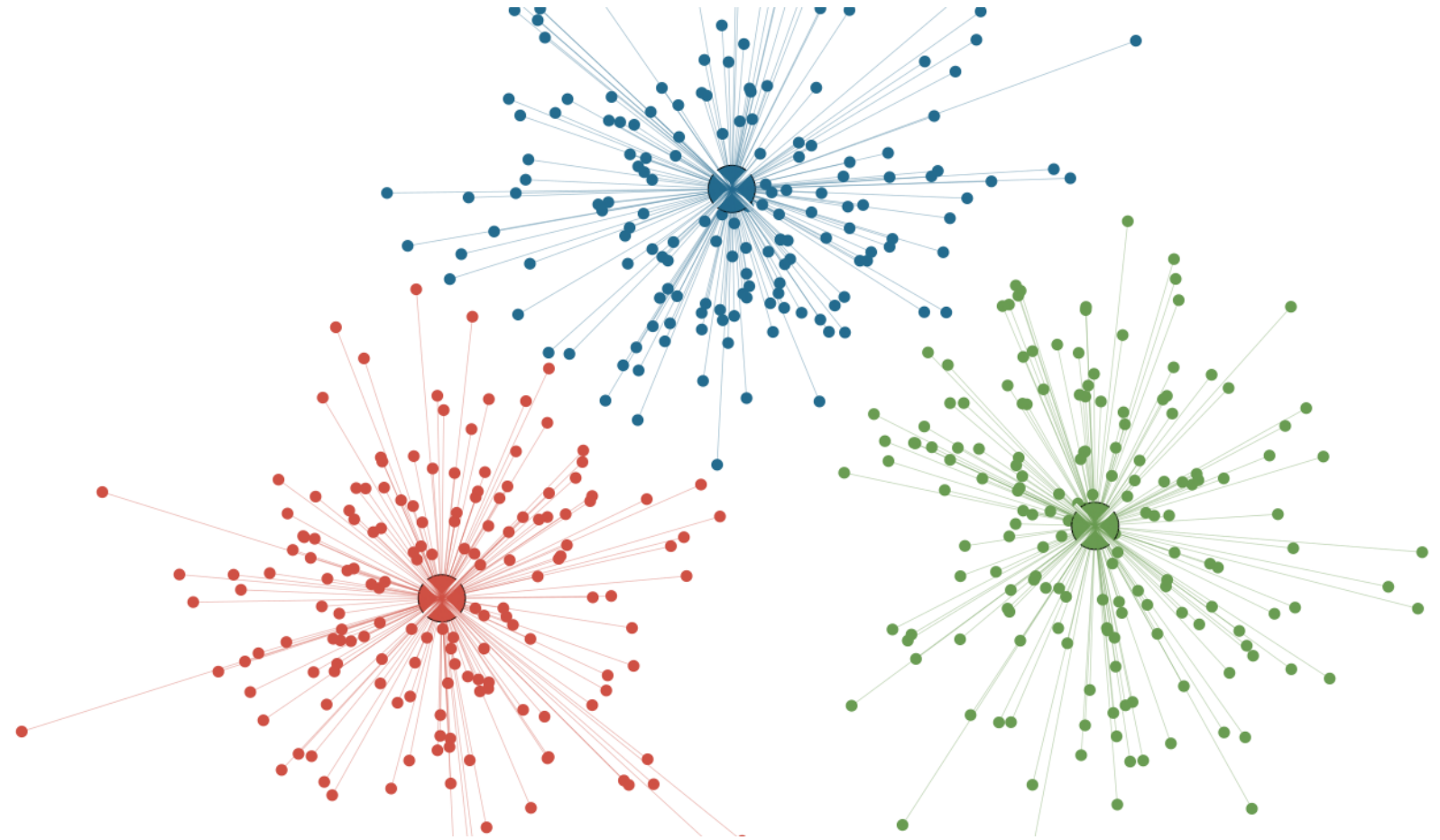




Segmentering med k-means

Et let tilgængeligt supplement til Matematik A-undervisningen på gymnasiet og handelsskolen med forklarende videoer og eksempler.

Jesper N. Wulff, Professor

Aarhus Universitet · Institut for Økonomi

2025, 1. udgave

Indhold

1	Kundesegmentering – når matematik møder virkeligheden	2
1.1	Eksempler på kundesegmenter	2
1.2	Fra intuition til algoritme	4
1.3	Afstand mellem punkter	5
1.4	K-means – trin for trin	8
1.4.1	Et eksempel på opdatering af en centroide.	9
1.4.2	Hvornår stopper k-means? (stoppebetingelsen)	9
1.5	Hvor mange klynger skal vi vælge?	11
1.6	Ekstra: Hvorfor kvadrerer vi afstandene?	16
2	K-means i praksis – byg din egen model i Excel	17
3	Opgaver	19
3.1	Diskussion	19
3.2	Øvelser – Basis	20
3.3	Øvelser – Udvidet	21
3.4	Excel	22

1 Kundesegmentering – når matematik møder virkeligheden

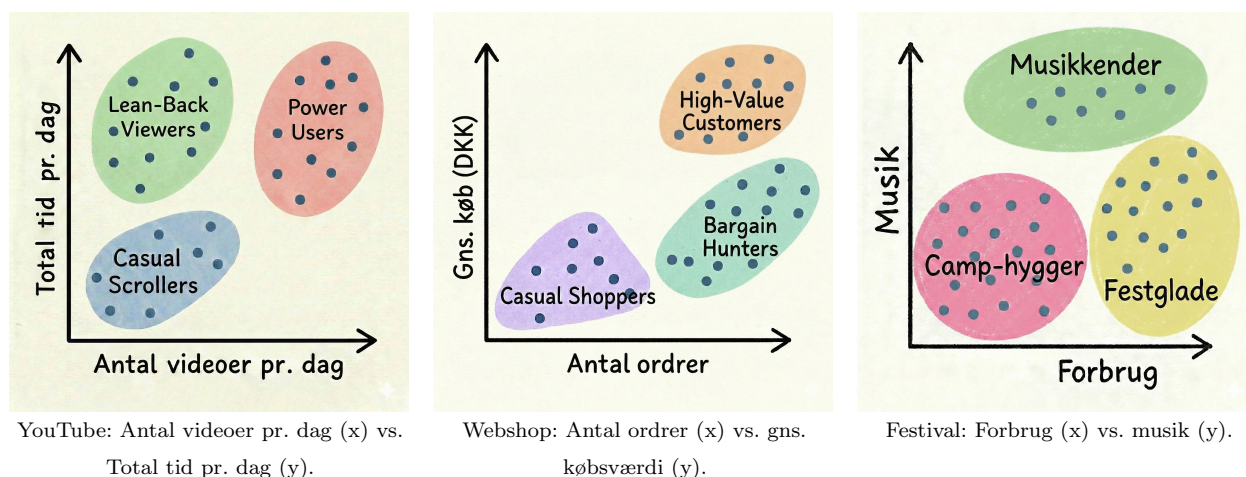
1.1 Eksempler på kundesegmenter

For at forstå idéen i kundesegmentering tager vi tre konkrete eksempler fra virkeligheden. I alle tre handler det om at bruge *mønstre i data* til at gruppere personer, som ligner hinanden, så indhold, tilbud og kommunikation kan målrettes bedre.

Eksempel 1 – YouTube: Hvad bruges alle de data til? Når du bruger YouTube, registreres bl.a. seertid, hvor mange videoer du ser, hvilke videoer du klikker på, tidspunkt på dagen, og hvilken enhed du bruger. Samlet set tegner disse signaler et stabilt mønster af seervaner. Nogle brugere ser mange korte klip: de hopper hurtigt videre, ser ofte Shorts og highlights, og har relativt kort seertid pr. session. Disse **Casual Scrollers** bruger YouTube i små pauser og reagerer stærkt på thumbnails og hurtige hooks. Andre bruger platformen mere afslappet og fokuseret: de ser få, men lange videoer ad gangen, ofte podcasts, dokumentarer eller lange essays. Disse **Lean-Back Viewers** bruger YouTube som en slags tv og bliver i samme indhold længe. En tredje gruppe er de mest engagerede: **Power Users**, som både ser mange videoer og bruger lang tid pr. session. De følger bestemte creators tæt, vender ofte tilbage og interagerer mere med indholdet. Resultatet er, at to personer på samme platform møder vidt forskellige forsider: anbefalinger, autoplay og annoncer justeres løbende, fordi adfærden placerer dem i forskellige klynger af seere.

Eksempel 2 – Webshoppen: Hvem skal have hvilket tilbud? Forestil dig en webshop, der analyserer kundedadfærd ud fra to enkle mål: hvor ofte der købes, og hvor meget der bruges pr. ordre. Over tid vil kunderne samle sig i genkendelige mønstre. **Casual Shoppers** køber sjældent og for relativt små beløb; de dukker typisk op, når et konkret behov opstår, og glemmer hurtigt webshoppen igen uden en påmindelse. **Bargain Hunters** køber ofte, men billigt: de reagerer stærkt på rabatter, kampagner og bundttilbud, og deres samlede volumen kan være høj, selvom marginen pr. køb er lav. Endelig er der **High-Value Customers**, som både køber ofte og bruger mange penge pr. ordre. De er loyale, mindre prisfølsomme og står for en stor del af omsætningen. Ved at identificere disse klynger kan webshoppen målrette indsatsen: friktionsfri genaktivering af de sjældne kunder, kampagner og upsell til de prisfølsomme, og VIP-behandling, personlig kommunikation og tidlig adgang til de mest værdifulde.

Eksempel 3 – Musikfestivalen: Hvem kommer – og hvad vil de? Tænk på en festival, hvor vi kigger på to enkle kendetegn: tid foran scenen og forbrug i boderne. Her opstår tre tydelige grupper. **Camp-hyggerne** bliver mest i lejren og bruger relativt lidt – de trives, når festivalen kommer til dem. **De festglade** er meget ved scenerne, mødes med mange og køber mere – de søger fællesskab og energi. **Musikkenderne** bruger længst tid foran scenen, men har mindre fokus på forbrug – de jagter lyd, nærvær og viden. Samme festival, men tre forskellige rejser; og netop derfor giver det mening at planlægge, kommunikere og prissætte forskelligt.



Figur 1: Tre eksempler på kundesegmentering. Få, enkle målte kendetegn kan allerede afsløre naturlige klynger.

Med disse grupperinger kan man begynde at udvikle målrettede tiltag. På YouTube betyder det, at anbefalinger, autoplay og annoncetyper skifter: Casual Scrollers møder korte klip og brede annoncer, Lean-Back Viewers fastholdes i lange serier og tematiske playlists, og Power Users eksponeres for nicheindhold og community-funktioner. I webshoppen kan High-Value Customers tilbydes VIP-fordele og personlig service, Bargain Hunters mødes med kampagner og bundles, og Casual Shoppers får en enkel og venlig vej tilbage. På festivalen kan Musikkenderne tiltrækkes med artist-talks og programdybde, Camp-hyggerne med stemningsskabende events uden for festivalsområdet, og de Festglade med pakker, der kombinerer koncerter, mad og premium-områder. Samme data – tre forskellige måder at skabe værdi på.

Video 1 – Klynger og segmenter

Varighed: 6 minutter

Indhold: I Video 1 gennemgår vi tre eksempler på kundesegmentering og ser på:

- Hvordan YouTube, en webshop og en festival kan segmentere deres kunder.
- Hvordan man med kundesegmentering kan rette tiltag til hvert segment.
- Hvordan man kan forestille sig kunder som punkter i et koordinatsystem.

Men hvordan finder vi egentlig frem til disse grupper, når vi står med tusindvis af personer og ingen forhåndsviden om, hvem der ligner hinanden? Det er netop her, data og algoritmer kommer os til hjælp. Ved at lade en computer lede efter mønstre i data kan vi automatisk opdage grupper af personer, der minder om hinanden. I stedet for at gætte os frem til segmenterne, kan vi *lade data fortælle os*, hvordan de naturligt opstår. I det næste afsnit ser vi derfor, hvordan en simpel men effektiv metode – **k-means algoritmen** – kan opdele et stort datasæt i mindre, men meningsfulde grupper, hvor observationerne inden for hver gruppe ligner hinanden mest muligt.

1.2 Fra intuition til algoritme

I eksemplerne kunne vi tydeligt se de forskellige grupper. Punkterne er af et overskueligt antal og pænt adskilt på tværs af de to akser. Men YouTube har milliarder af brugere, der kan karakteriseres ved ikke ét, men hundredevis af forskellige kendetegn. Det gør det umuligt for os mennesker at danne os et overblik ved at kigge på en simpel graf. Vi kan ganske enkelt ikke længere “se” mønstrene.

Det er her, algoritmer kommer ind i billedet. En algoritme er en opskrift – en præcis og gentagelig måde at løse et problem på. I vores tilfælde ønsker vi en algoritme, der automatisk kan finde grupper af gæster, som ligner hinanden ud fra deres adfærd. Det er det, **k-means algoritmen** kan. Den tager alle punkterne (altså brugerne/kunderne/gæsterne) og leder efter mønstre, så hver person bliver placeret i den gruppe, som de passer bedst til.

Tænk på algoritmen som en hjælper, der prøver at “gætte”, hvordan du selv ville have grupperet gæsterne, hvis du havde haft tid og overblik. Den starter med et tilfældigt bud på, hvordan grupperne ser ud, og forbedrer derefter løsningen trin for trin. Til sidst ender den med en gruppering, hvor gæsterne inden for samme klynge ligner hinanden mest muligt – og samtidig adskiller sig mest muligt fra de andre klynger.

I næste afsnit skal vi se præcis, hvordan denne proces fungerer i praksis – trin for trin. Vi følger k-means-algoritmen fra den første tilfældige inddeling til de endelige, stabile klynger.

1.3 Afstand mellem punkter

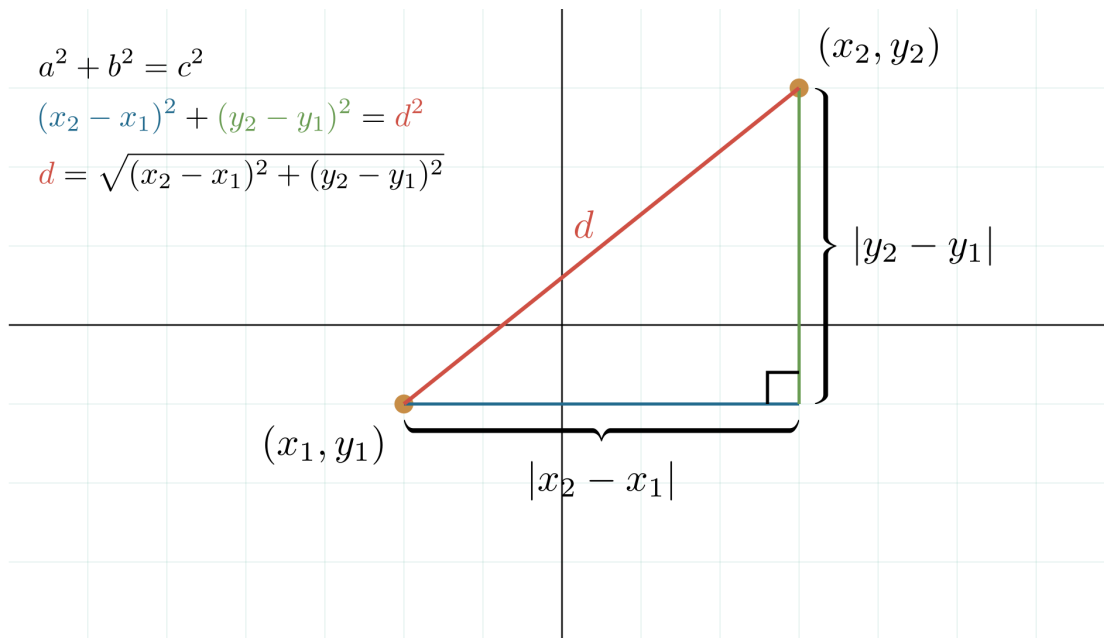
K-means-algoritmen er en metode, der automatisk opdeler data i grupper, hvor observationerne inden for hver gruppe ligner hinanden mest muligt. Den bruges i mange sammenhænge: fra festivalgæster og kundesegmenter til digitale anbefalingssystemer.

For at forstå k-means må vi forstå, hvordan vi afgør, om to punkter ligner hinanden. Vi siger, at to kunder / gæster / brugere ligner hinanden, når deres afstand er lille. Tag et kig på Figur 2 nedenfor, hvor idéen er gjort mere formel. Vi ønsker at finde ud af afstanden mellem to punkter $A(x_1, y_1)$ og $B(x_2, y_2)$. Ved at tegne en trekant, kan vi se, at problemet i virkeligheden er at finde længden på siden d , som er afstanden mellem de to punkter.

Afstanden mellem to punkter i planen beregnes ved hjælp af Pythagoras' sætning. Hvis vi har to punkter $A(x_1, y_1)$ og $B(x_2, y_2)$, er den euklidiske afstand:

$$d(A, B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}.$$

Dette udtryk er uvurderligt til at bestemme, hvilke datapunkter der ligner hinanden mest. Hvis x er *kroner* og y er *antal ordre*, så finder vi afstanden mellem de to kunder ved at tage forskellen mellem køb i kroner i anden plus forskellen i antal ordre i anden. Den samlede sum tager vi så kvadratroden af til sidst.



Figur 2: Afstanden mellem to punkter findes ud fra Pythagoras' sætning.

Lad os prøve at sætte tal ind i formlen. Antag, at vi har to punkter: $A(-2, -1)$ og $B(3, 3)$. Afstanden bliver:

$$d(A, B) = \sqrt{(3 - (-2))^2 + (3 - (-1))^2} = \sqrt{25 + 16} \approx 6.4.$$

Dette er vores mål for afstanden mellem de to punkter. Du kan forestille dig, at $x_1 = -2$ betyder, at person 1, P_1 , bruger 2 kr. mindre end gennemsnittet, og $y_1 = -1$ at P_1 lægger 1 ordre ind mindre end gennemsnittet. Så 0 er altså gennemsnittet, positive tal betyder *mere end gennemsnittet* og negative tal betyder *mindre end gennemsnittet*. Det afhænger helt af, hvad tallene refererer til i det datasæt, vi arbejder med.

Forestil dig, at der nu kommer data på en ny kunde, $P_{ny}(2, 0)$, som bruger 2 kroner mere i gennemsnittet med lige så mange ordre som gennemsnittet. Vi vil gerne finde ud af, hvem den nye kunde ligner mest. Vi udregner derfor afstanden til hver af de to gamle kunder. Lad os sætte tallene ind i formlen og udregne afstanden til P_{ny} og se, hvem der er tættest på.

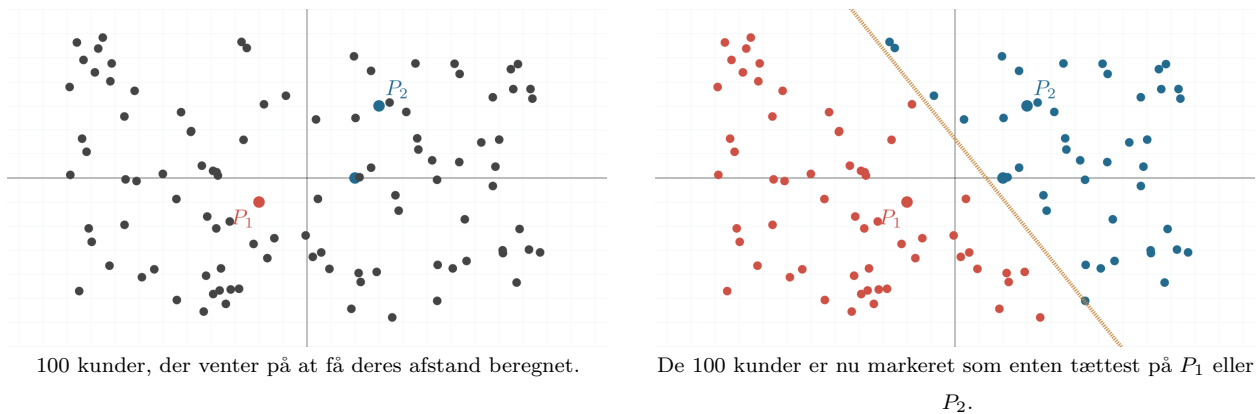
$$d_1 = \sqrt{(2 - (-2))^2 + (0 - (-1))^2} \approx 4.12$$

$$d_2 = \sqrt{(2 - 3)^2 + (0 - 3)^2} \approx 3.16$$

P_{ny} ligner altså P_2 mest, da afstanden til P_2 er kortere.

”Det var smart”, tænker du måske. ”Men hvad nu hvis vi havde 100 punkter?” Intet problem! Den helt store fordel ved at have lavet en formaliseret regel for, hvordan afstand måles, er, at vi nu kan lade computeren kværne igennem 1000-vis af punkter på et splitsekund.

I Figur 3 til venstre kan du se 100 forskellige punkter (kunder). Vi vil gerne finde ud af, om de hver især mest ligner P_1 eller P_2 . Vores regel siger, at hvis man er tættest på P_1 , så hører man til den røde klynge, og hvis man er tættest på P_2 , så hører man til den blå klynge. Til højre i Figur 3 vises punkterne nu opdelt i to klynger. Den gule streg viser beslutningsgrænsen - er man på den ene side, er man tættere på P_1 , og er man på den anden side, er man tættere på P_2 .



Figur 3: Fra datapunkter til klynger på et splitsekund

Video 2 – Hvad vil det sige, at to kunder ligner hinanden?

Varighed: 7 minutter

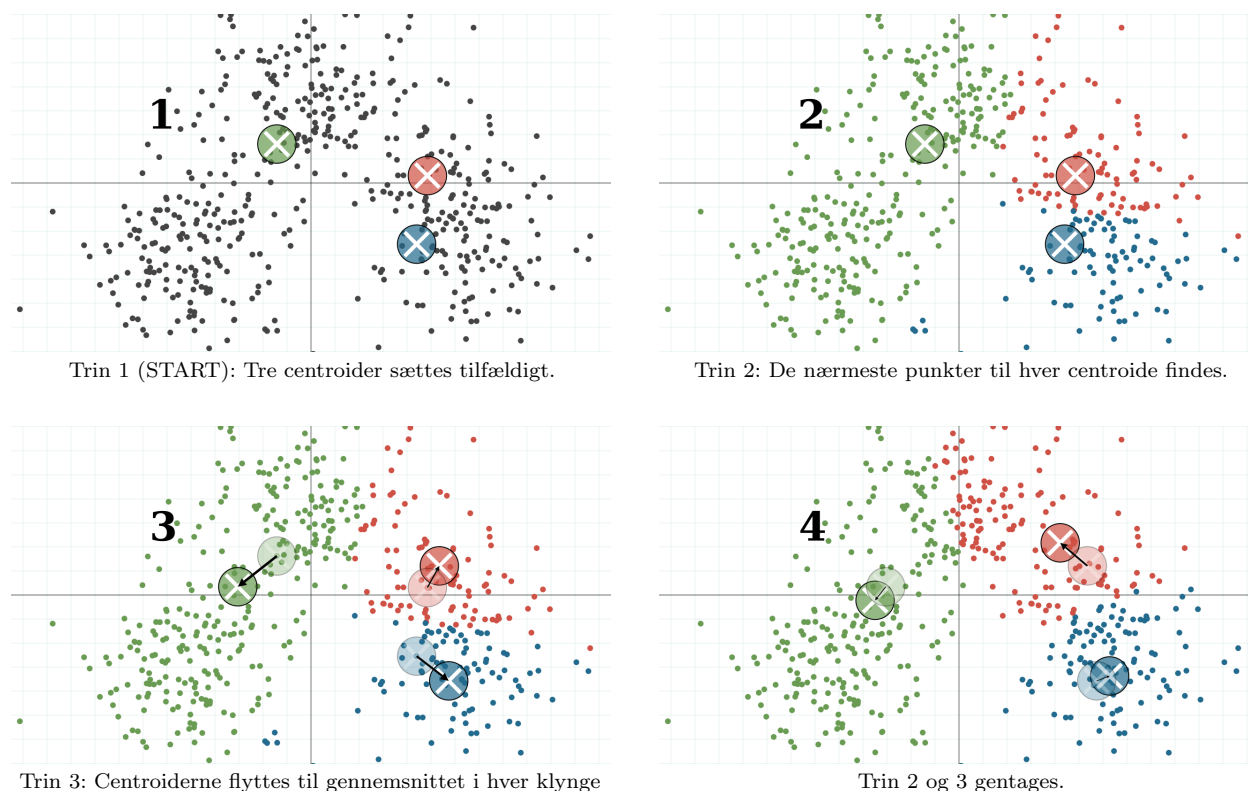
Se Video 2 for en gennemgang af:

- Hvordan vi kan finde afstanden mellem to punkter vha. pythagoras.
- Et numerisk eksempel på at finde afstanden mellem to punkter.
- Hvordan vi tildeler (mange) nye punkter til forskellige segmenter.

Vi har nu den helt afgørende ingrediens i hånden til at se og forstå opskriften på k-means algoritmen. Med en metode til at beregne, om to punkter (igen: tænk kunder) er tæt på (”ligner”) hinanden, kan vi nu gå videre med opskriften på k-means algoritmen.

1.4 K-means – trin for trin

Figur 4 viser, hvordan k-means skridt for skridt. Lad os gennemgå den ét skridt af gangen.



Figur 4: k-means trin for trin

Figur 4 viser, hvordan k-means virker skridt for skridt, når vi ønsker at opdele data i tre klynger.

Trin 1 – Start-centroider. Algoritmen starter med tre tilfældigt valgte centroider. I begyndelsen er de ikke nødvendigvis placeret tæt på de egentlige grupper, men vi skal starte et sted. Dette er vist i Figur 4 øverst til venstre.

Trin 2 – Tildel punkter til nærmeste centroide. Nu findes de punkter, der er tættest på hver centroide. Hvert punkt tildeles den klynge, hvis centrum det er tættest på. I Figur 4 (2) er det vist ved, at punkterne nu har samme farve som centroiden.

Trin 3 – Opdatér centroiderne som gennemsnit. Når alle punkter er tildelt en klynge, beregnes et nyt centrum for hver gruppe ved at tage gennemsnittet af punkterne i klyngen. Centroiden flyttes derefter mod midten af gruppen som vist i Figur 4 (3). Derefter opdateres alle punkternes tilhørsforhold. Nogle punkter flytter nu klynge, da de ligger tættere på en anden centroide.

Trin 2 og 3 gentages. Processen gentages. Nogle punkter skifter side, og centroiderne flytter sig lidt igen. Hver gang vi udfører trin 2 og 3, siger vi, at vi har kørt en iteration. Figur 4 (4) viser resultatet af at køre én iteration mere, altså udføre trin 2 og 3 igen.

Stop. Til sidst ændres grupperne ikke længere. Klyngerne er nu så homogene som muligt — punkterne inden for hver klynge ligner hinanden meget, mens der er større forskel på punkterne i forskellige klynger. Algoritmen er *konvergeret*.

Tillykke - du kender nu k-means algoritmen! Lad os kigge på to taleksempler for at sikre os, at vi er helt med. Først kigger vi på opdateringen af centroiden og dernæst på stoppebetingelsen.

1.4.1 Et eksempel på opdatering af en centroide.

Når punkterne er tildelt klynger, flyttes hvert centrum til midten af de punkter, der hører til den. Det betyder, at vi beregner gennemsnittet af punkternes koordinater i klyngen.

Antag, at klynge 1 består af tre punkter:

$$(2, 4), \quad (3, 5), \quad (4, 3)$$

For at finde det nye centrum beregner vi gennemsnittet af x-koordinaterne og y-koordinaterne hver for sig:

$$x_{\text{centrum}} = \frac{2 + 3 + 4}{3} = 3, \quad y_{\text{centrum}} = \frac{4 + 5 + 3}{3} = 4.$$

Det nye centrum for klyngen bliver derfor punktet:

$$(3, 4)$$

Dette punkt repræsenterer midten af gruppen og bliver den nye **centroide**. I næste iteration vil afstanden fra alle datapunkter igen blive målt ud fra denne opdaterede position.

Disse to trin – afstand og opdatering – gentages, indtil centroiderne næsten ikke flytter sig mere.

1.4.2 Hvornår stopper k-means? (stoppebetingelsen)

K-means-algoritmen gentager de samme to trin igen og igen: (1) *tildel punkter til nærmeste center* og (2) *opdatér centroiderne som gennemsnit*. Men hvornår giver det mening at stoppe denne proces?

Den mest almindelige stoppebetingelse bygger på en enkel idé: **Vi stopper, når centroiderne næsten ikke flytter sig længere.**

Man vælger derfor en lille grænseværdi, ofte kaldet ε (fx $\varepsilon = 0,01$). Hvis hvert center flytter sig mindre end ε fra én iteration til den næste, betragtes løsningen som stabil, og algoritmen stoppes.

Lad os prøve at sætte nogle tal på i et simpelt eksempel: Antag, at en centroid i iteration 3 ligger i punktet

$$C^{(3)} = (0,50; 0,40)$$

og efter næste opdatering ligger i

$$C^{(4)} = (0,52; 0,41).$$

Flytningen måles som afstanden mellem de to positioner:

$$\text{flyt} = \sqrt{(0,52 - 0,50)^2 + (0,41 - 0,40)^2} = \sqrt{0,02^2 + 0,01^2} \approx 0,022.$$

Hvis stoppegrænsen er sat til $\varepsilon = 0,01$, gælder

$$0,022 > 0,01,$$

og vi fortsætter derfor algoritmen — centroiden flytter sig stadig *mærkbart*.

Antag nu, at næste opdatering giver

$$C^{(5)} = (0,521; 0,409).$$

Flytningen bliver da

$$\sqrt{(0,521 - 0,52)^2 + (0,409 - 0,41)^2} \approx 0,0014.$$

Nu gælder

$$0,0014 < 0,01,$$

så centroiden flytter sig ikke længere ret meget. Når dette gælder for *alle* centroider, siger vi, at k-means er *konvergeret*, og algoritmen stopper. Intuitionen er, at centroiderne flytter sig meget i starten, fordi grupperne endnu ikke er på plads. Vi satte jo bare punkterne tilfældigt, så vi skal være meget heldige, hvis vi rammer tæt på den mest optimale opdeling. Efterhånden som klyngerne stabiliseres, bliver justeringerne mindre og mindre — til sidst er der ingen grund til at fortsætte.

Video 3 – K-means trin for trin

Varighed: 7 minutter

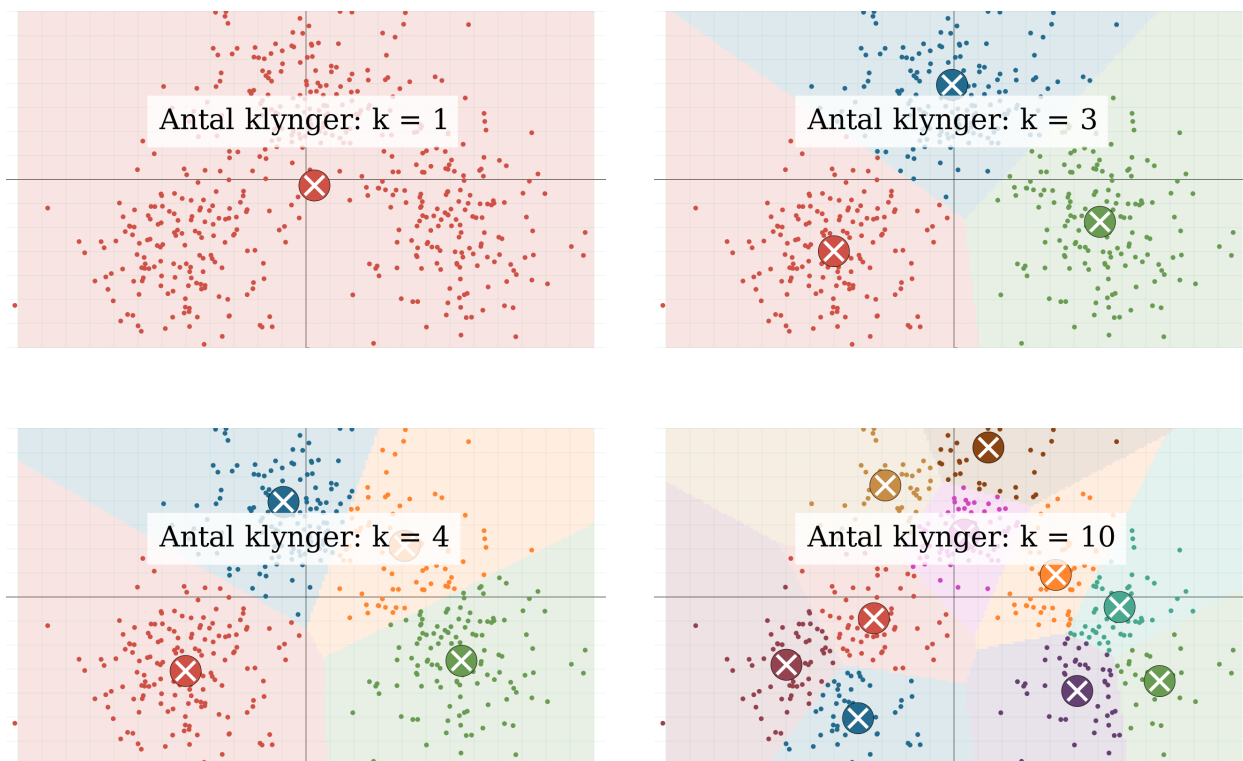
Se Video 3 for en gennemgang af:

- Hvordan k-means starter (Trin 1).
- Hvordan k-means tilføjer punkter til klynger (Trin 2).
- Hvordan k-means flytter centroider til klyngegegnsmittet (Trin 3)
- Hvordan k-means gentager Trin 2 og 3 (itererer) indtil centroiderne ikke bevæger sig længere (stoppebetingelsen).

1.5 Hvor mange klynger skal vi vælge?

Indtil nu har vi taget det for givet, at der findes tre klynger. Men i virkeligheden ved vi sjældent på forhånd, hvor mange grupper (k) der findes. Skal festivalgæsterne, webshopkunderne og youtube-brugerne opdeles i to, tre eller flere segmenter?

I figur 5 ser du et eksempel, hvor vi har opdelt de samme data i hhv. 1, 3, 4 og 10 klynger. K-means-modellen ændrer sin struktur markant, når antallet af klynger k øges. Ved $k = 1$ behandles alle observationer som én samlet gruppe. Det kunne vi have gjort uden brug af en avanceret algoritme. Det kaldes også **underfitting** - vi har lært for lidt af data. Ved $k = 3$ opstår en naturlig og intuitiv opdeling, der afspejler de underliggende strukturer i data. Flere klynger end dette (f.eks. $k = 4$ eller $k = 10$) skaber stadig mindre forbedringer, men til gengæld også en mere kompleks model. Vælger vi en klyngeopdeling med for høj kompleksitet, risikerer vi, at vores model ikke generaliserer ud over det pågældende datasæt. Det kaldes **overfitting**. Med andre ord får vi eks. 10 klynger, som ikke svarer til den endelige opdeling af kunder i den virkelige verden, men 10 klynger, som kun findes i det her datasæt.

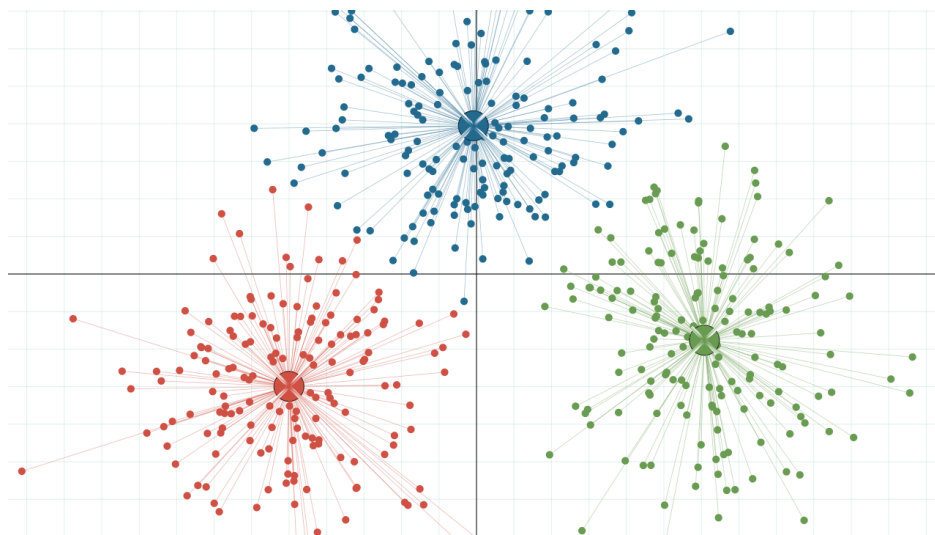


Figur 5: Et spørgsmål om balance: Én klynge er for lidt, 10 er for mange, 3 er lige tilpas. Hvert panel viser det *endelige resultat* for et givent k (øverst fra venstre: $k = 1, 3$; nederst: $k = 4, 10$). Når k øges, falder inert i (punkterne kommer tættere på et centrum), og grænserne tilpasser sig mere detaljeret til data. Samtidig bliver modellen mindre overskuelig. Figuren illustrerer den centrale trade-off: *simpel* model med få klynger versus *detaljeret* model med mange klynger.

Kort sagt: Det handler om balance. Jo flere klynger vi vælger, desto bedre kan modellen “passe” data, fordi hver gruppe bliver mindre og mere præcis. I yderste konsekvens kunne vi vælge k lige så stort som antallet af observationer, så hver person får sin egen klynge. Det ville give perfekt tilpasning, men nul indsigt – vi har ikke lært noget generelt, kun fininddelt det pågældende datasæt. Med andre ord er vores klyngemodell maksimalt overfittet.

Derfor handler valget af k om at finde en balance: *simpel nok til at give overblik – men detaljeret nok til at fange de vigtigste forskelle*. I praksis bruger man ofte en grafisk metode kaldet “**albue-metoden**”. Her måler man, hvor meget den samlede variation inden for klyngerne falder, når man øger antallet af klynger.

Først måler man afstanden inden for hver klynge, ligesom vi har gjort indtil videre vha. Pythagoras. Denne afstandsmåling er visualiseret i Figur 6 nedenfor. Herefter kvadreres afstandene og lægges sammen, så vi har én samlet værdi. Denne værdi kaldes *inerti*, og det er et mål for, hvor



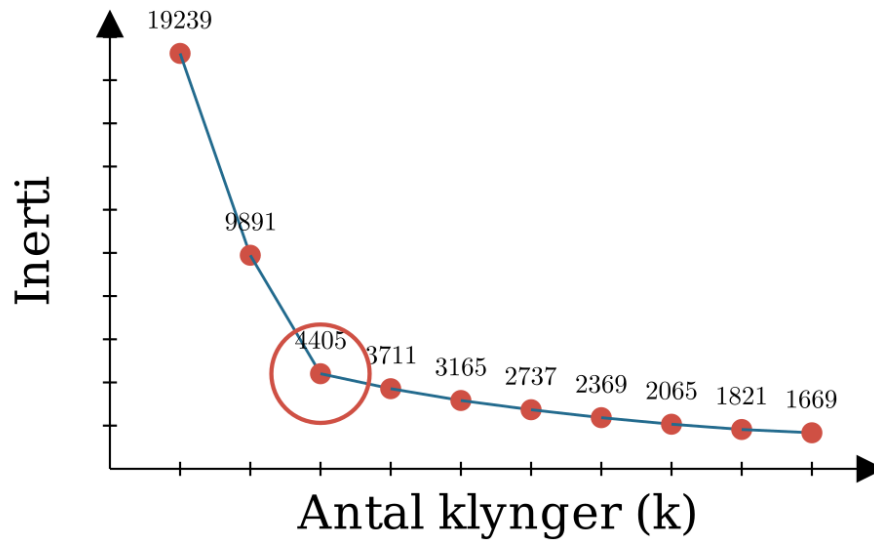
Figur 6: Første skridt til at bestemme antallet af klynger er at udregne inerti. Inerti er summen af de kvadrerede afstande fra hvert punkt til den centroide, som punktet hører til. I figuren ses netop disse afstande.

langt punkterne samlet set ligger fra deres klyngemidte (centroide).

Næste skridt er at lave et **albueplot**. Det gør man sådan her:

1. Beregn summen af kvadrerede afstande fra hvert punkt til sin centroide (kaldet *inerti*) for forskellige værdier af k .
2. Tegn et plot med k på x-aksen og inerti på y-aksen.
3. Kurven falder hurtigt i starten, men flader derefter ud – som en arm med en albue.
4. Punktet hvor kurven “knækker” (albuen) viser ofte det mest naturlige valg af k .

I Figur 7 kan vi se et eksempel på et albueplot. I begyndelsen falder inertien kraftigt, men efter $k = 3$ flader kurven ud. Punktet ved $k = 3$ kaldes *albuen*, og det indikerer, at yderligere klynger kun forbedrer modellen minimalt i forhold til den ekstra kompleksitet. Dette tyder på, at tre klynger er et passende valg for datasættet. Vælger vi $k < 3$ resulterer det i underfit, hvor vi lærer for lidt af data, mens et $k > 3$ resulterer i overfit, hvor vi lærer for meget af data.



Figur 7: **Albue-metoden.** Figuren viser, hvordan den samlede inerti (summen af kvadrerede afstande fra observationer til deres nærmeste centrum) falder, når antallet af klynger k øges.

Et simpelt eksempel: Én vs. to klynger. For at forstå, hvorfor albue-metoden virker, kan vi se på et meget lille og konkret eksempel. Antag, at vi har seks datapunkter, som fordeler sig sådan her:

$$(1; 1), (2; 1), (1; 2) \quad \text{og} \quad (8; 8), (9; 8), (8; 9)$$

Punkterne ligger tydeligt i to grupper — tre tæt på $(1; 1)$ og tre tæt på $(8; 8)$. Lad os sammenligne, hvad der sker, hvis algoritmen tvinges til at samle alle punkter i *én* klynge ($k = 1$), versus når den får lov til at danne *to* klynger ($k = 2$).

1) Én klynge ($k = 1$)

Når der kun er én klynge, bliver centroiden midtpunktet for alle seks punkter:

$$x_{\text{centrum}} = \frac{1 + 2 + 1 + 8 + 9 + 8}{6} = 4,83, \quad y_{\text{centrum}} = \frac{1 + 1 + 2 + 8 + 8 + 9}{6} = 4,83.$$

Centroiden ligger altså i punktet $(4,83; 4,83)$. Vi beregner nu de kvadrerede afstande fra hvert punkt til centrum:

$$\begin{aligned}
(1; 1) : & (1 - 4,83)^2 + (1 - 4,83)^2 = 29,4 \\
(2; 1) : & (2 - 4,83)^2 + (1 - 4,83)^2 = 22,7 \\
(1; 2) : & (1 - 4,83)^2 + (2 - 4,83)^2 = 22,7 \\
(8; 8) : & (8 - 4,83)^2 + (8 - 4,83)^2 = 20,1 \\
(9; 8) : & (9 - 4,83)^2 + (8 - 4,83)^2 = 27,4 \\
(8; 9) : & (8 - 4,83)^2 + (9 - 4,83)^2 = 27,4
\end{aligned}$$

Den samlede **inerti** (summen af de kvadrerede afstande) bliver derfor:

$$\text{Inerti}_{k=1} = 29,4 + 22,7 + 22,7 + 20,1 + 27,4 + 27,4 = 149,67$$

2) To klynger ($k = 2$)

Nu opdeler vi punkterne i to naturlige grupper:

- Klynge 1: $(1; 1), (2; 1), (1; 2) \rightarrow$ centrum $(1,33; 1,33)$
- Klynge 2: $(8; 8), (9; 8), (8; 9) \rightarrow$ centrum $(8,33; 8,33)$

Afstandene til de nærmeste centroider bliver nu meget små:

$$\begin{aligned}
(1; 1) : & (1 - 1,33)^2 + (1 - 1,33)^2 = 0,22 \\
(2; 1) : & (2 - 1,33)^2 + (1 - 1,33)^2 = 0,56 \\
(1; 2) : & (1 - 1,33)^2 + (2 - 1,33)^2 = 0,56 \\
(8; 8) : & (8 - 8,33)^2 + (8 - 8,33)^2 = 0,22 \\
(9; 8) : & (9 - 8,33)^2 + (8 - 8,33)^2 = 0,56 \\
(8; 9) : & (8 - 8,33)^2 + (9 - 8,33)^2 = 0,56
\end{aligned}$$

Summen af de kvadrerede afstande bliver nu:

$$\text{Inerti}_{k=2} = 0,22 + 0,56 + 0,56 + 0,22 + 0,56 + 0,56 = 2,67$$

Sammenligning og fortolkning

Antal klynger (k)	Centroid-placeringer	Samlet inerti
$k = 1$	Ét centrum $(4,8; 4,8)$	149,67
$k = 2$	To centre $(1,3; 1,3)$ og $(8,3; 8,3)$	2,67

Når vi går fra $k = 1$ til $k = 2$, falder inertien dramatisk fra omkring 149,6 til blot 2,67. Det viser, hvor meget bedre to klynger beskriver data end én stor fælles klynge. Hvis man fortsætter til $k = 3$ eller flere, falder inertien kun en lille smule — forbedringen flader ud. Det er netop dér, “albuen” i albue-metoden opstår.

Video 4 – Hvordan vælger vi antallet af klynger?

Varighed: 12 minutter

Se Video 4 for en gennemgang af:

- Hvad der sker, hvis vi hæver eller sænker k .
- Hvad det vil sige at under- og overfitte.
- Hvordan vi udregner inertien.
- Hvordan vi kan bruge inertien og albueplottet til at vælge antallet af klynger.

1.6 Ekstra: Hvorfor kvadrerer vi afstandene?

Hvorfor kvadrerede vi egentligt afstandene, da vi udregnede inertien? Lad os prøve at give lidt intuition gennem et simpelt eksempel:

Forestil dig, at tre punkter ligger omkring deres centrum:

$$-3, \quad 0, \quad +3.$$

Hvis vi bare lægger afvigelseerne sammen, får vi

$$(-3) + 0 + (+3) = 0,$$

så det ser ud, som om punkterne ligger helt samlet – men de ligger jo spredt ud på hver sin side!

Ved i stedet at kvadrere afvigelseerne bliver alle positive:

$$(-3)^2 + 0^2 + (+3)^2 = 9 + 0 + 9 = 18.$$

Summen fortæller nu, hvor stor den samlede spredning er. Det samme princip gælder i k-means: vi summerer de *kvadrerede afstande* fra hvert punkt til sit centrum for at måle, hvor tæt punkterne

ligger samlet.

Derfor: Kvadrering sikrer, at både “venstre” og “højre” afvigelser tæller positivt med i inertien — og at punkter langt væk vægter ekstra højt.

Video 5 - (EKSTRA): Hvorfor kvadrerer vi afstande

Varighed: 4 minutter

Indhold: I Video 5 kan du se en gennemgang af:

- tre måder at beregne afstande på.
- et taleksempel, der illustrerer, hvordan måderne virker.

I næste afsnit skal vi se, hvordan vi kan implementere k-means i Excel og undersøge, hvordan algoritmen faktisk udfører sine beregninger i praksis.

2 K-means i praksis – byg din egen model i Excel

Vi har nu forstået idéen bag k-means: at finde k midtpunkter, der bedst repræsenterer vores data. Nu skal vi prøve at gøre det selv — direkte i Excel, uden brug af avanceret software. Det giver en klar fornemmelse af, hvordan algoritmen rent faktisk arbejder, når den opdaterer klynger og centroider.

Trin for trin: Byg din egen k-means i Excel

1. **Data:** Åbn excel-filen k_means.k2.xlsx.
2. **Start-centroider:** Vælg 2 tilfældige punkter som start (fx skriv dem manuelt i kolonne H-I).
3. **Afstande:** Beregn afstanden mellem hvert punkt og hver centroide med Pythagoras' formel, eks. for C1:
$$=SQRT((A2-H\$5)^2+(B2-H\$6)^2)$$
4. **Tilknyt klynge:** Brug =IF() til at afgøre, hvilken centroide der er tættest på:
$$=IF(C2<D2;1;2)$$
5. **Opdater centroider:** Beregn gennemsnit af hver gruppes punkter med =AVERAGEIF():
$$=AVERAGEIF(\$E\$2:\$E\$501;1;A2:A501)$$
6. **Gentag:** Kopiér de nye centroider tilbage til startcellerne og gentag beregningen, indtil tildelingene ikke længere ændrer sig. Husk at bruge "paste values" (indsæt værdier).

Video 6 - K means i Excel

Varighed: 10 minutter

Indhold: Video 6 demonstrerer trin for trin, hvordan k-means implementeres i Excel:

- Hvordan man indtaster data og vælger start-centroider
- Hvordan man beregner afstande mellem datapunkter og centroider
- Hvordan man tildeler punkterne til deres nærmeste klynge
- Hvordan man opdaterer centroiderne som gennemsnit af klyngerne
- Hvordan processen gentages, indtil der ikke længere sker ændringer

Faldgruber

- **Glemte opdateringer:** Hvis du ikke kopierer de nye centroider tilbage til start, itererer modellen ikke korrekt.
- **Antal klynger:** Husk, at k skal vælges på forhånd — brug albue-metoden som rettesnor.
- **Målestoksproblemer (normalisering):** Hvis dine variable måles i meget forskellige enheder — fx *forbrug i kroner* og *aktivitet i timer* — vil den med de største tal dominere afstandsberegningen. For at undgå dette kan man **normalisere** hver variabel, så alle måles på samme skala.

Sådan gør du i Excel (engelske funktionsnavne):

$$x' = \frac{x - \text{AVERAGE}(x)}{\text{STDEV}(x)}$$

Det kan implementeres direkte i Excel med funktionen `STANDARDIZE()`:

```
=STANDARDIZE(A2; AVERAGE($A$2:$A$731); STDEV($A$2:$A$731))
```

Funktionen trækker gennemsnittet fra hver observation og dividerer med standardafvigelsen. Efter normalisering har alle variable middelværdi 0 og standardafvigelse 1, så afstandene beregnes retfærdigt — ingen variabel “vinder”, bare fordi den måles i større tal.

3 Opgaver

3.1 Diskussion

Opgave 1 (lighed). Hvad vil det sige, at to punkter ligner hinanden i k-means? Nævn mindst to mulige lighedsmål.

Opgave 2 (algoritme). Beskriv trinene i k-means fra start til stop. Hvorfor kan forskellige start-centroider give forskellige løsninger?

Opgave 3 (inerti). Hvad måler *inerti*, og hvorfor (typisk) falder den fra iteration til iteration?

Opgave 4 (valg af k). Hvad sker der, hvis man vælger et alt for stort k ? Nævn mindst to ulemper.

3.2 Øvelser – Basis

Opgave 5 (afstand). Brug de to punkter $A(2;3)$ og $B(5;7)$. Beregn den euklidiske afstand $d(A, B)$.

Opgave 6 (kvadreret afstand). To kunder har $(x, y) = (320; 2,1)$ og $(290; 1,9)$. Et center står i $(300; 2,0)$. Beregn de *kvadrerede* afstande til centret. Hvem er nærmest?

Opgave 7 (opdatering af center). Du har tre punkter i klynge A: $(2;3)$, $(4;5)$, $(3;4)$. Hvor flytter center A hen efter én opdatering?

Opgave 8 (inerti). For punkterne $(1;1)$, $(2;1)$, $(2;2)$ og center $(1,5; 1,5)$: beregn inertien (summen af kvadrerede afstande til centret).

Opgave 9 (beslutningsgrænse i 2D). Vi har to centre $C_1 = (0;0)$ og $C_2 = (4;0)$.

1. Bestem den lodrette linje, der udgør beslutningsgrænsen (hvor et punkt er lige langt fra de to centre).
2. Klassificér punktet $P = (1;3)$: hører det til klynge 1 eller 2?

Opgave 10 (én iteration, $k = 2$). Vi har fire punkter:

$$P_1 = (0;0), \quad P_2 = (0;2), \quad P_3 = (4;0), \quad P_4 = (4;2).$$

Start-centroider: $C_1^{(0)} = (0;0)$ og $C_2^{(0)} = (4;2)$.

1. Tildel hvert punkt til nærmeste centroid.
2. Opmåler centroiderne til $C_1^{(1)}$ og $C_2^{(1)}$.
3. Beregn inertien efter opdateringen (brug de nye centroider og de tilhørende klynger).

Opgave 11 (effekt af skala). Vi har tre kunder med to variable: forbrug (kr.) og antal køb:

$$A = (1000;1), \quad B = (1200;2), \quad C = (4000;3).$$

1. Beregn afstanden $d(A, B)$ og $d(A, C)$ (euklidisk).
2. Forklar, hvilken variabel der dominerer afstandene og hvorfor.

3. (Bonus) Hvis man dividerer forbrug med 1000 (så enheden bliver “tusinde kr.”), hvad sker der så med sammenligningen mellem afstandene?

3.3 Øvelser – Udvidet

Opgave 12 (2 iterationer, $k = 2$). Vi har seks punkter:

$$(1; 1), (2; 1), (1; 2), (8; 8), (9; 8), (8; 9).$$

Start-centroider: $C_1^{(0)} = (1; 2)$ og $C_2^{(0)} = (9; 8)$.

1. Udfør én iteration: tildel klynger og opdatér til $C_1^{(1)}, C_2^{(1)}$.
2. Udfør en iteration mere og få $C_1^{(2)}, C_2^{(2)}$.
3. Er algoritmen konvergeret efter 2 iterationer? Begrund kort.

Opgave 13 (tiebreak). Et punkt P ligger præcis lige langt fra to centre.

1. Giv et konkret eksempel med tal (et punkt og to centre), hvor afstandene bliver ens.
2. Nævn to mulige regler (tiebreakers) for, hvad man gør i praksis.
3. Diskutér kort, om tiebreakeren kan påvirke den endelige løsning.

Opgave 14 (inerti før vs. efter opdatering). Vi har tre punkter i én klynge:

$$(0; 0), (0; 2), (2; 0).$$

1. Beregn inertien hvis centret er sat til $C = (0; 0)$.
2. Beregn inertien hvis centret i stedet er sat til gennemsnittet af punkterne.
3. Hvad konkluderer du om, hvorfor gennemsnittet er et “godt” center i k-means?

Opgave 15 (outlier-effekt). Vi har punkterne:

$$(0; 0), (0; 1), (1; 0), (1; 1)$$

i én klynge.

1. Find centroiden og inertien.
2. Tilføj et outlier-punkt (20; 20) til samme klynge. Find den nye centroid.
3. Forklar (kort) hvad outliers gør ved centroider og inerti.

3.4 Excel

Opgave 16 (k=2 i Excel). Åbn excel-filen `k_means_k2.xlsx`. Udfyld kolonnerne “Afstand C1”, “Afstand C2” og “Klynge”.

Opgave 17 (konvergens). Vælg et konvergenskriterium (fx 0,01). Udfør algoritmen og tæl, hvor mange iterationer der er nødvendige for konvergens. Gentag ved forskellige startværdier.

Opgave 18 (k=3 i Excel). Åbn excel-filen `k_means_k3.xlsx`. Udfyld kolonnen “Afstand C3”, cellerne “P12” og “P13”, og den totale ændring i celle “M9”.

Opgave 19 (konvergens). Vælg et konvergenskriterium (fx 0,01). Udfør algoritmen og tæl, hvor mange iterationer der er nødvendige for konvergens. Gentag ved forskellige startværdier.

Opgave 20 (inerti). Tjek inerti og sammenlign med to-klynge løsningen. Gentag ved forskellige startværdier.