Strojno učenje

13. Grupiranje

prof. dr. sc. Bojana Dalbelo Bašić
doc. dr. sc. Jan Šnajder

Sveučilište u Zagrebu
Fakultet elektrotehnike i računarstva

Ak. god. 2012/13.

v1.1

Danas...

1. Nenadzirano učenje
2. Grupiranje
3. Algoritam k-srednjih vrijednosti
4. Hijerarhijsko grupiranje
5. Provjera grupa

Danas...

Nenadzirano učenje

U mnogim slučajevima ne znamo koji primjer pripada kojoj klasi.
Umjesto skupa označenih primjera:

$$\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$$

x_1	x_2	$\ldots$	x_n	y
$x_1^{(1)}$	$x_2^{(1)}$	$\ldots$	$x_n^{(1)}$	$y^{(1)}$
$x_1^{(2)}$	$x_2^{(2)}$	$\ldots$	$x_n^{(2)}$	$y^{(2)}$
$\vdots$	$\vdots$		$\vdots$	$\vdots$
$x_1^{(N)}$	$x_2^{(N)}$	$\ldots$	$x_n^{(N)}$	$y^{(N)}$

raspolažemo skupom neoznačenih primjera (engl. *unlabeled data*):

$$\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$$

Nenadzirano učenje

Primjeri su neoznačeni jer:

(1) ne znamo ih označiti (ne znamo unaprijed koje klase postoje)

(2) označavanje je preskupo

Npr.

- Grupiranje klijenata prema ponašanju (segmentacija korisnika)
- Grupiranje novinskih članaka prema temama/autorima
- Grupiranje gena sa sličnom izražajnošću (funkcionalnošću)
- Klasifikacija objekata na fotografiji (*content-based image retrieval*)
- Klasifikacija Twitter-poruka prema iskazanom raspoloženju
- Otkrivanje čudnog ponašanja korisnika na mreži (*intrusion detection*)

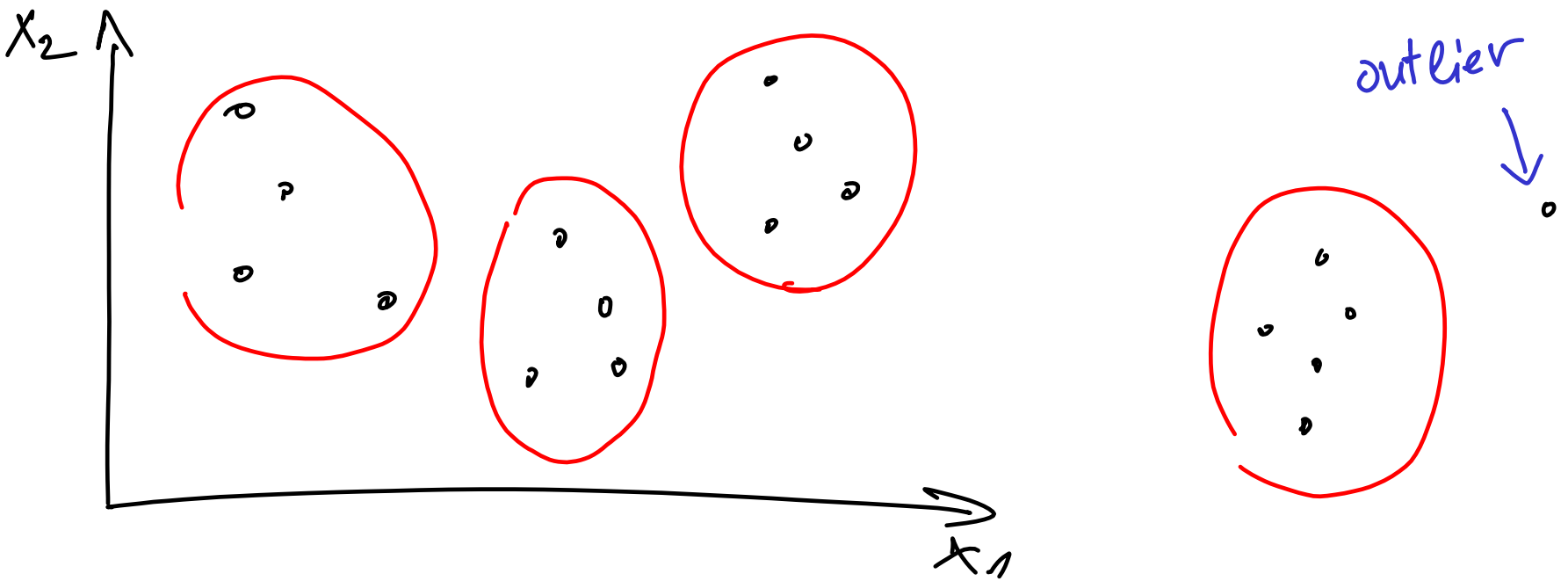
Nenadzirano učenje

Tri zadatka:

(1) grupiranje (engl. *clustering*)

(2) otkrivanje novih vrijednosti ili vrijednosti koje odskaču (engl. *novelty/outlier detection*)

(3) smanjenje dimenzionalnosti (engl. *dimensionality reduction*)



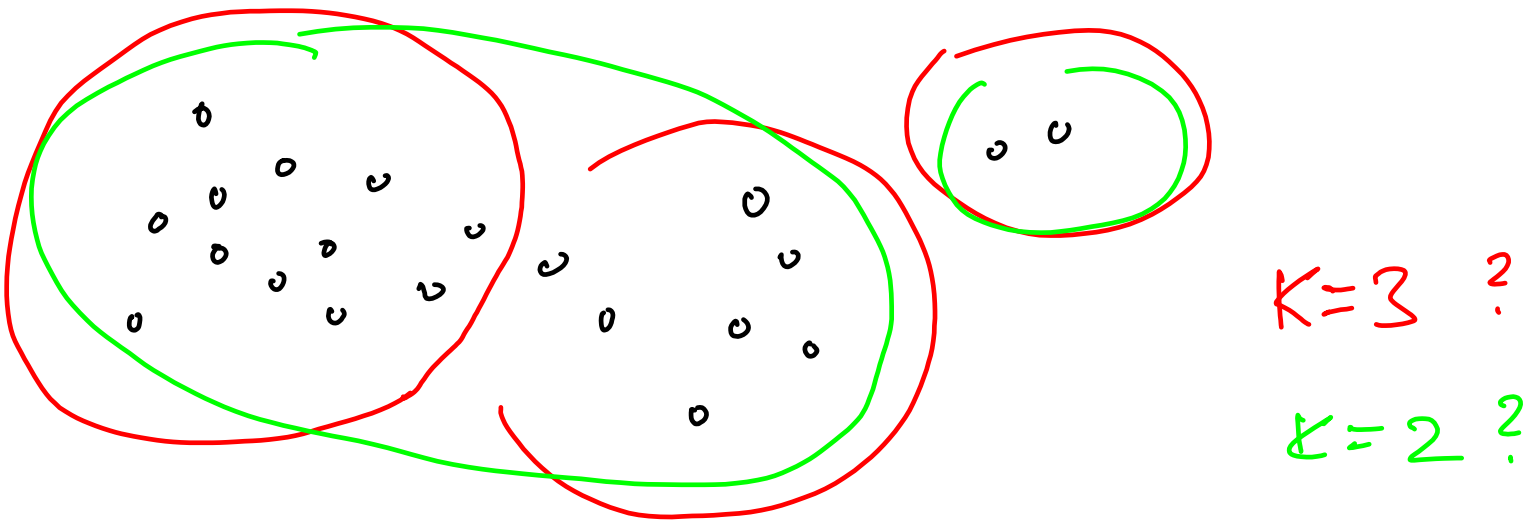
Danas. . .

Grupiranje

Grupiranje (engl. *clustering*)

Razdjeljivanja primjera u grupe (engl. *clusters*) primjera, tako da su slični primjeri (slični po nekom svojstvu) svrstani u istu grupu, a različiti primjeri u različite grupe.

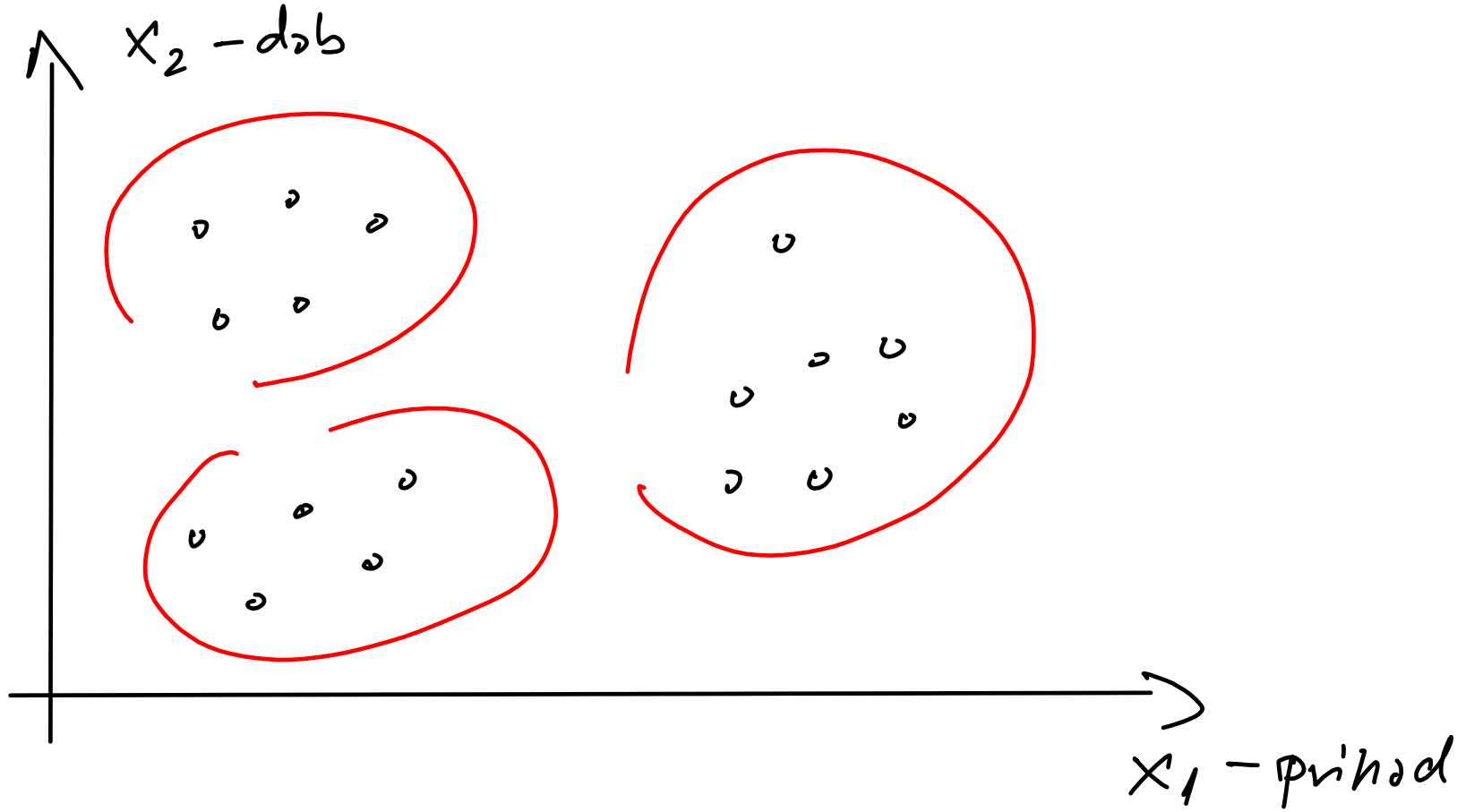
Svrha grupiranja jest nalaženje "prirodnih" (intrinzičnih) grupa u skupu neoznačenih podataka. (*Let the data speak for itself!*)



Primjer 1

Grupiranje korisnika prema prihodu i dobi

$\mathbf{x} = (x_1, x_2)$, x_1 – prihod, x_2 – dob



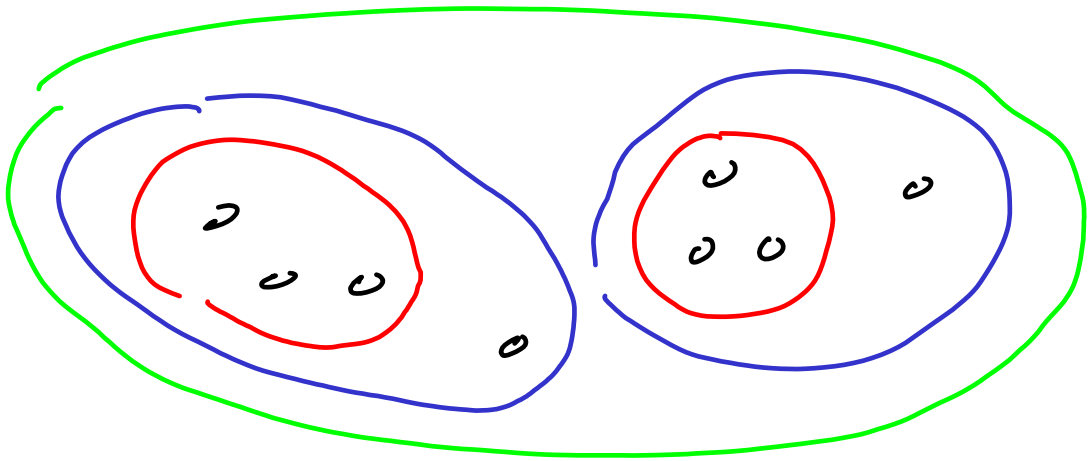
Primjer 2

Grupiranje programskih kôdova po sličnosti (*code-clone detection*)

$\mathbf{x} = (x_1, \ldots, x_n)$?

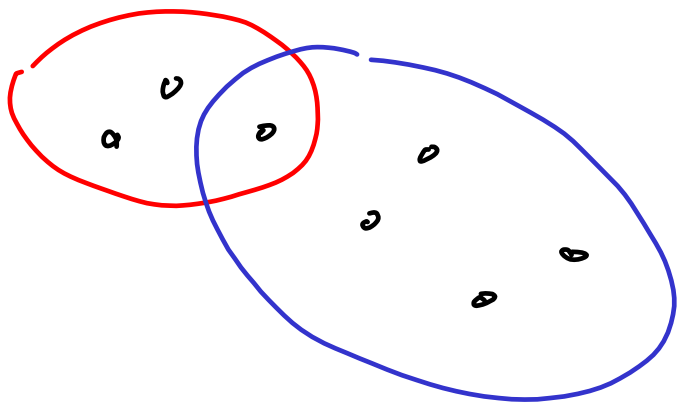
Vrste grupiranja

(1) particijsko grupiranje
(2) hijerarhijsko grupiranje



(1) čvrsto grupiranje
(2) meko grupiranje – primjer može pripadati u više grupa

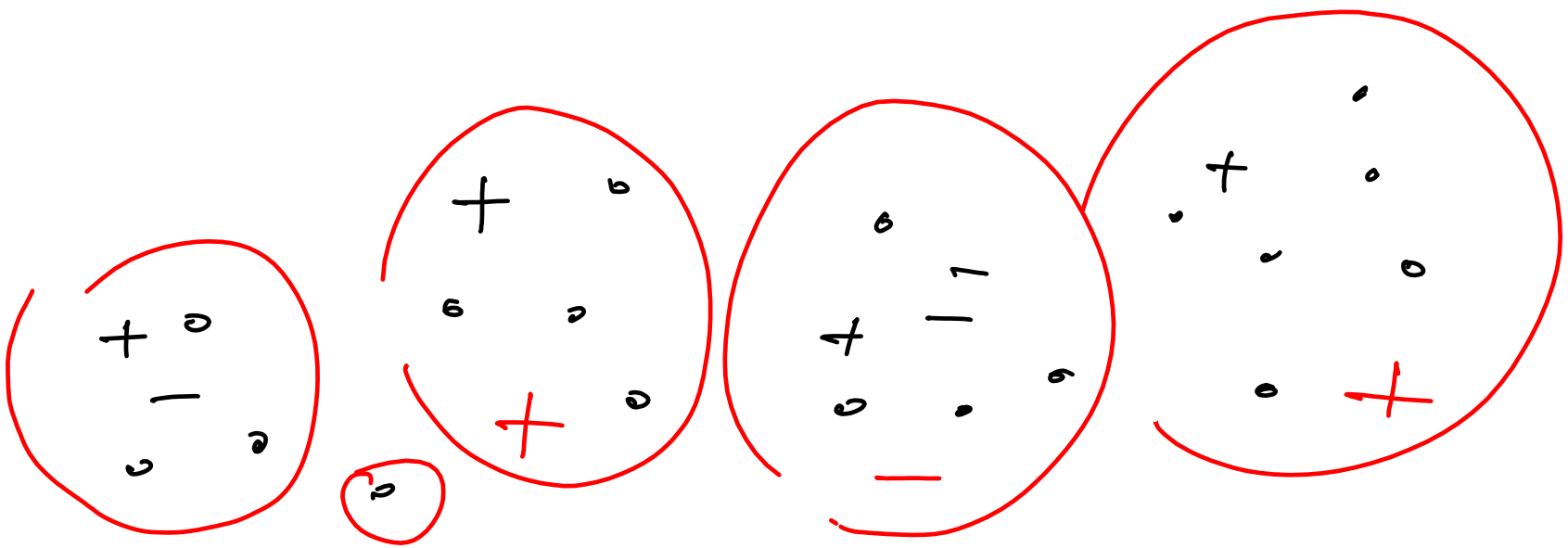
↳ fuzzy k-means
↳ probabilističko grupiranje (EM-algoritam)



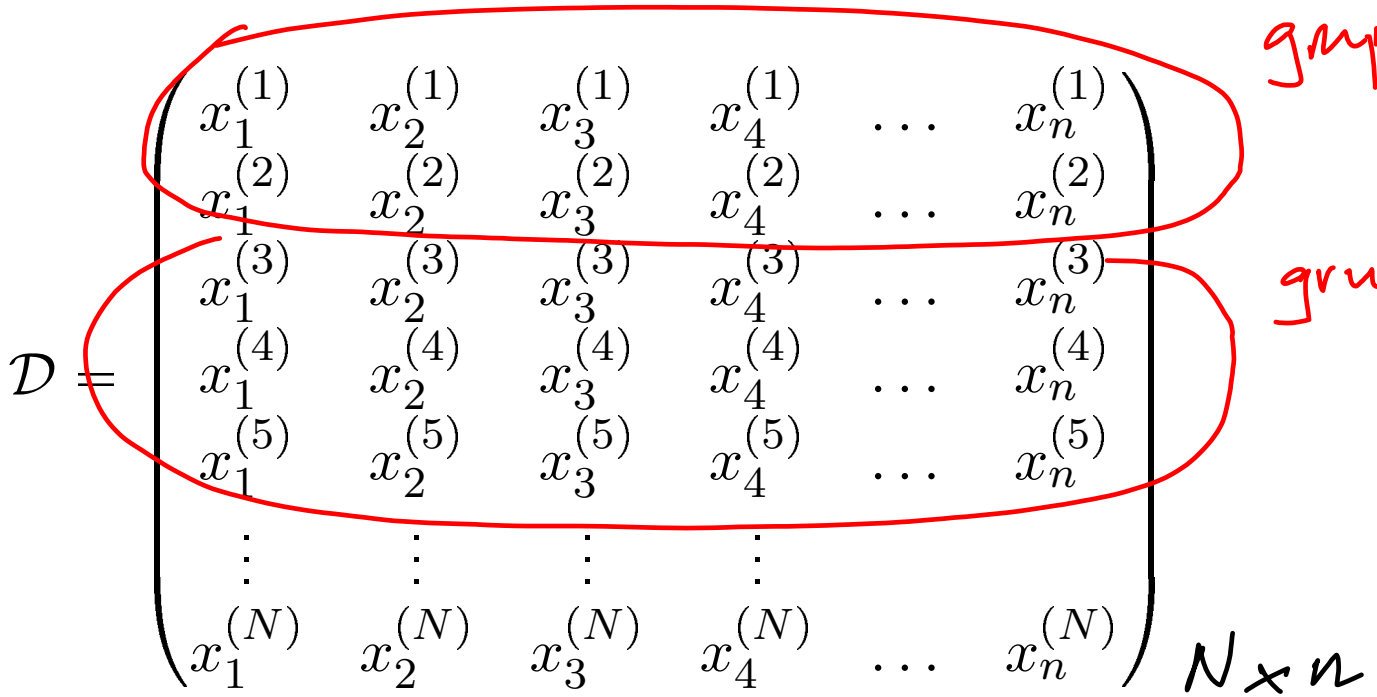
Primjena grupiranja

- istraživanje podataka (engl. *data exploration*)
- kompresija podataka
- predobrada za klasifikaciju/regresiju: smanjenje broja primjera/značajki
- polunadzirano učenje (semi-supervised learning)

↳ "cluster & label"



Grupiranje primjera



Smanjenje matice $N \times n \rightarrow K \times n$, $K < N$

=> zaglađivanje primjera (smoothing)

Grupiranje značajki

$$\mathcal{D} = \begin{pmatrix} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & x_4^{(1)} & \dots & x_n^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & x_4^{(2)} & \dots & x_n^{(2)} \\ x_1^{(3)} & x_2^{(3)} & x_3^{(3)} & x_4^{(3)} & \dots & x_n^{(3)} \\ x_1^{(4)} & x_2^{(4)} & x_3^{(4)} & x_4^{(4)} & \dots & x_n^{(4)} \\ x_1^{(5)} & x_2^{(5)} & x_3^{(5)} & x_4^{(5)} & \dots & x_n^{(5)} \\ \vdots & \vdots & \vdots & \vdots & & \\ x_1^{(N)} & x_2^{(N)} & x_3^{(N)} & x_4^{(N)} & \dots & x_n^{(N)} \end{pmatrix}$$

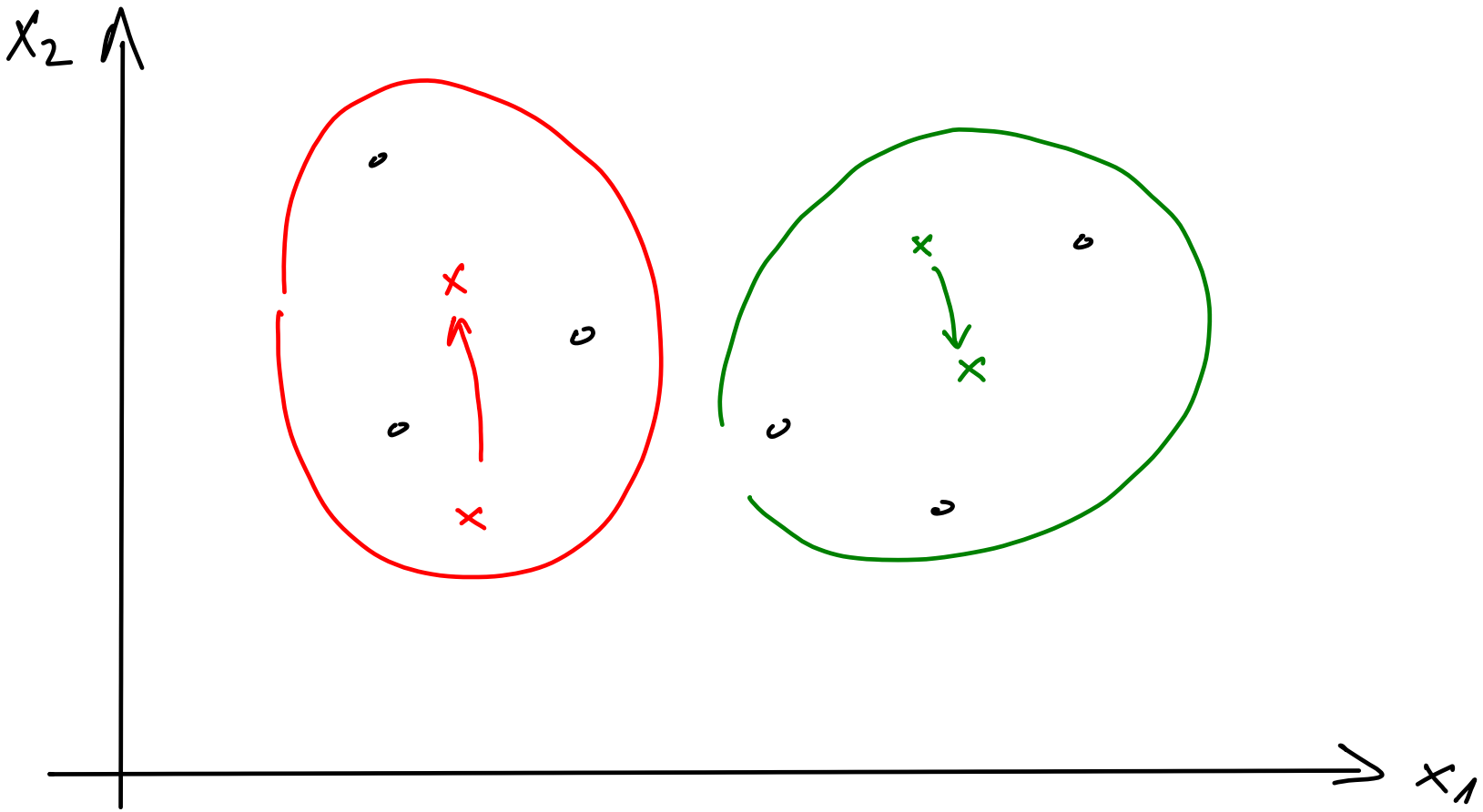
Smanjenje matrice $N \times n \longrightarrow N \times k$, $k < n$

Danas. . .

Algoritam k-srednjih vrijednosti

Particijsko grupiranje u K grupa. K je unaprijed određen!

Ideja: svaka grupa ima svoju srednju vrijednost (centroid). Svaki primjer pripada grupi čiji mu je centroid najbliži (po euklidskoj udaljenosti).



Algoritam k-srednjih vrijednosti

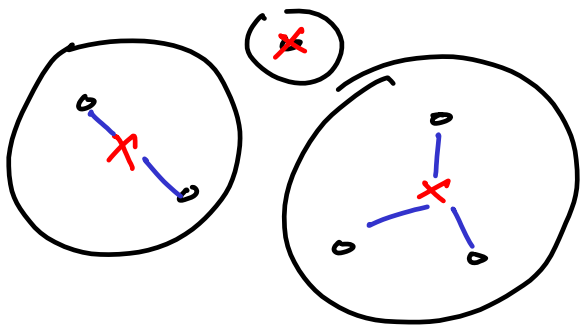
Kriterijska funkcija (funkcija pogreške):

indikatorska varijabla

$$J = \sum_{k=1}^{K} \sum_{i=1}^{N} b_k^{(i)} \|\mathbf{x}^{(i)} - \boldsymbol{\mu}_k\|^2$$

zbroj kv. udaljenosti od centroida

$$\|\mathbf{x} - \boldsymbol{\mu}\|^2 = (\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}} (\mathbf{x} - \boldsymbol{\mu})$$



Želimo grupiranje koje minimizira pogrešku:

$$\underset{b_1,\ldots,b_K;\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_K}{\operatorname{argmin}} \quad J$$

Analitička minimizacija ∇J po oba parametara nije moguća jer su međusobno ovisni.

Algoritam k-srednjih vrijednosti

(1) Slučajeno odaberi početne centroide $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$

(2) Pogrešku minimiziramo ako primjere razvrstamo u najbliže grupe:

$$b_k^{(i)} = \begin{cases} 1 & \text{ako } k = \underset{j}{\operatorname{argmin}} \|\mathbf{x}^{(i)} - \boldsymbol{\mu}_j\| \\ 0 & \text{inače} \end{cases}$$

(3) Uz fiksirane vrijednosti $b_k^{(i)}$, minimizacijom dobivamo nove centroide grupa:

$$\nabla_{\boldsymbol{\mu}_k} J = 2 \sum_{i=1}^{N} b_k^{(i)} (\mathbf{x}^{(i)} - \boldsymbol{\mu}_k) = \mathbf{0} \quad \Rightarrow \quad \boldsymbol{\mu}_k = \frac{\sum_i b_k^{(i)} \mathbf{x}^{(i)}}{\sum_i b_k^{(i)}}$$

(4) Novi centroidi su dobiveni uz fiksirane $b_k^{(i)}$. Promjenom centroida, neki $b_k^{(i)}$ su se promijenili, pa treba ponoviti od koraka (2).

Algoritam k-srednjih vrijednosti

Algoritam k-srednjih vrijednosti (engl. *k-means algorithm*)

inicijaliziraj centroide $\boldsymbol{\mu}_k$, $k = 1, \ldots, K$
ponavljaj
  za svaki $\mathbf{x}^{(i)} \in \mathcal{D}$
    $b_k^{(i)} \leftarrow \begin{cases} 1 & \text{ako } k = \operatorname{argmin}_j \|\mathbf{x}^{(i)} - \boldsymbol{\mu}_j\| \\ 0 & \text{inače} \end{cases}$
  za svaki $\boldsymbol{\mu}_k$, $k = 1, \ldots, K$
    $\boldsymbol{\mu}_k \leftarrow \sum_{i=1}^{N} b_k^{(i)} \mathbf{x}^{(i)} / \sum_{i=1}^{N} b_k^{(i)}$
dok $\boldsymbol{\mu}_k$ ne konvergiraju

Vremenska složenost: $\mathcal{O}\big(T(nNK + nN)\big) = \mathcal{O}(TnNK)$

Q: Je li algoritam deterministčan?

Svojstva algoritma

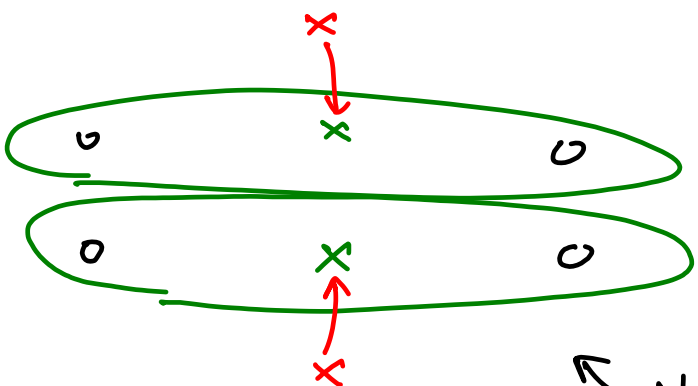
Algoritam zapravo pretražuje prostor stanja. Različitih stanja ima K^N.

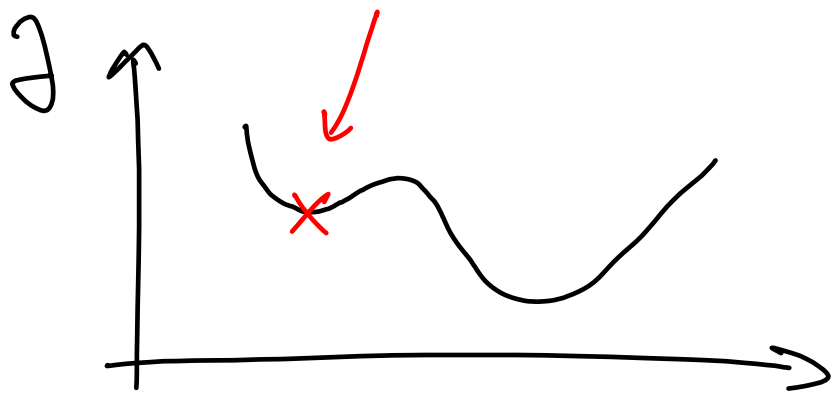
(1) Konvergira li algoritam?

Da, jer je broj konfiguracija konačan (konfiguracija = particija primjera + raspored središta) , a J se smanjuje kroz iteracije, pa algoritam nikada ne posjećuje istu konfiguraciju dva puta.

(2) Nalazi li algoritam optimalnu particiju?

Ne, jer je algoritam pohlepan i nalazi lokalno optimalno rješenje. Ishod ovisi o odabiru početnih središta.

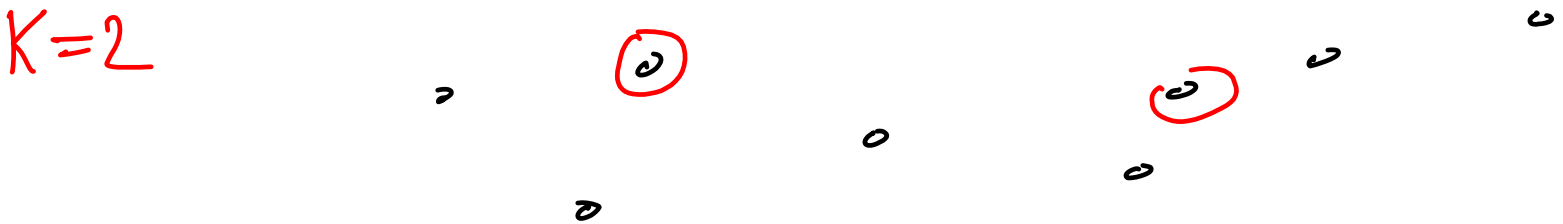




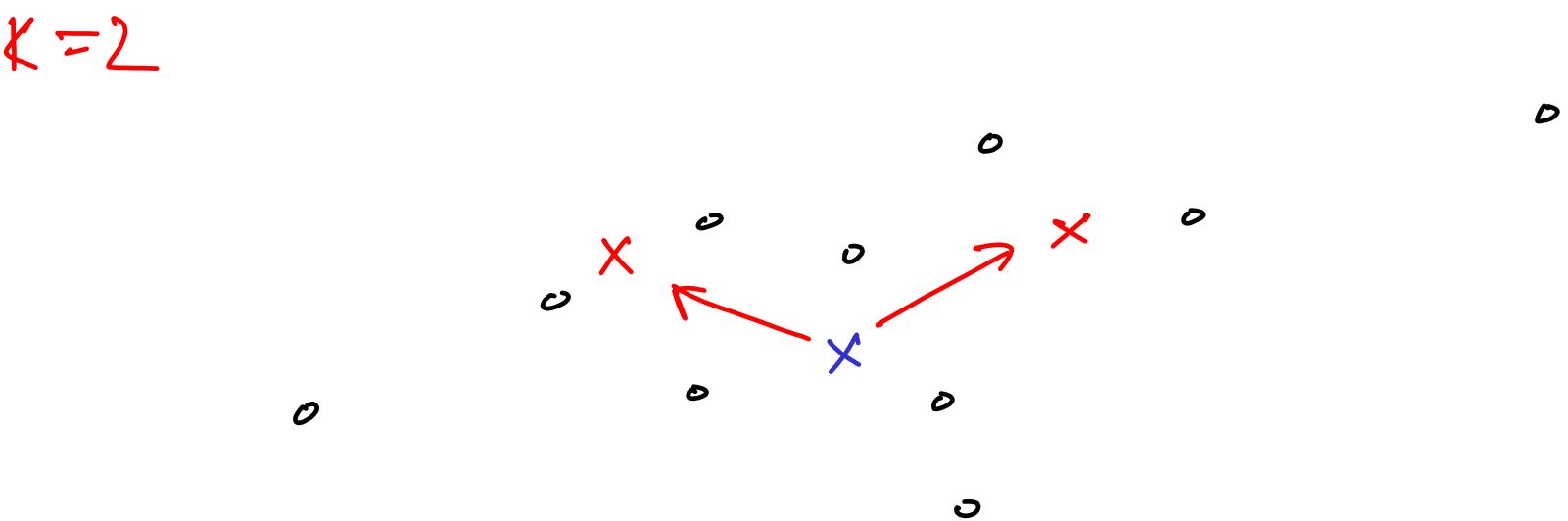
Neoptimalno rješenje

Odabir početnih središta

- nasumično odabrati K primjera kao središta (problem: vrijednosti koje odskaču)

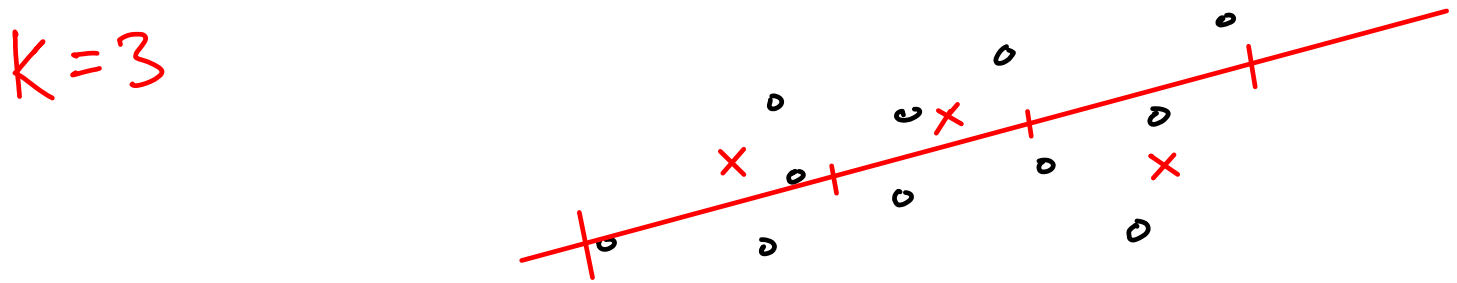


- nadodati slučajne vektore na centroid skupa $\mathcal{D}$



Odabir početnih središta

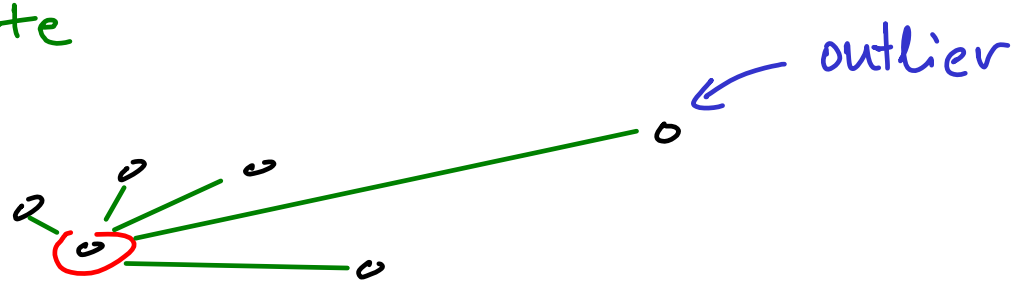
- razdijeliti prvu glavnu komponentu (PCA) u K jednakih intervala i uzeti središta primjera iz tih intervala



- **kmeans++**: slučajno odabrati početno središte, a zatim svako iduće središte odabrati tako da je što dalje od postojećih središta

$$P(\boldsymbol{\mu}_{k+1} = \mathbf{x}^{(i)} | \mathcal{D}, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k) = \frac{\min_k \|\boldsymbol{\mu}_k - \mathbf{x}^{(i)}\|^2}{\sum_j \min_k \|\boldsymbol{\mu}_k - \mathbf{x}^{(j)}\|^2}$$

novo središte



Algoritam k-medoida

Nekada primjere ne možemo prikazati u vektorskom prostoru.
Npr. grupiranje sličnih riječi ili grupiranje ljudi na temelju poznanstva.

Raspolažemo samo mjerom sličnosti/različitosti.

Definira matricu sličnosti: $S = \begin{pmatrix} 1 & 0,2 & 0,3 \\ 0,2 & 1 & 0,5 \\ 0,3 & 0,5 & 1 \end{pmatrix}$ (x^1, x^2, x^3)

Poopćenje algoritma k-srednjih vrijednosti: algoritam k-medoida (PAM algoritam).

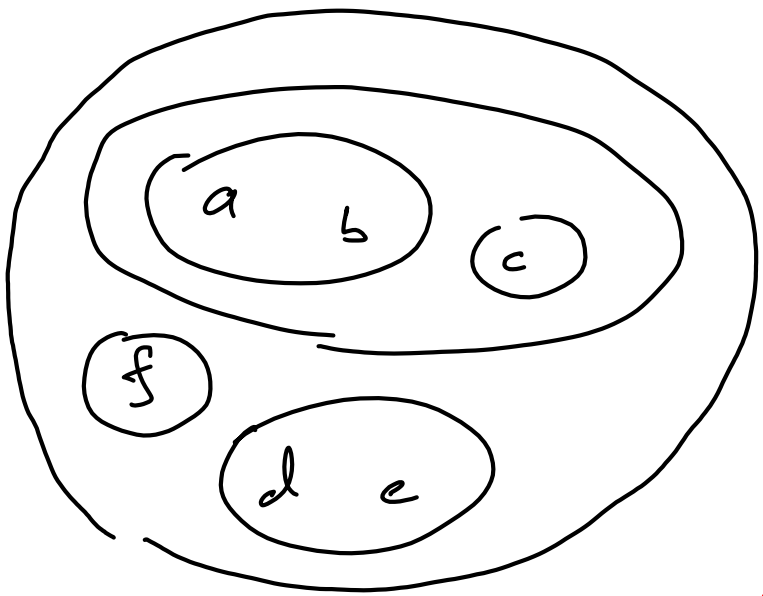
$\Rightarrow$ pogledati u skripti!

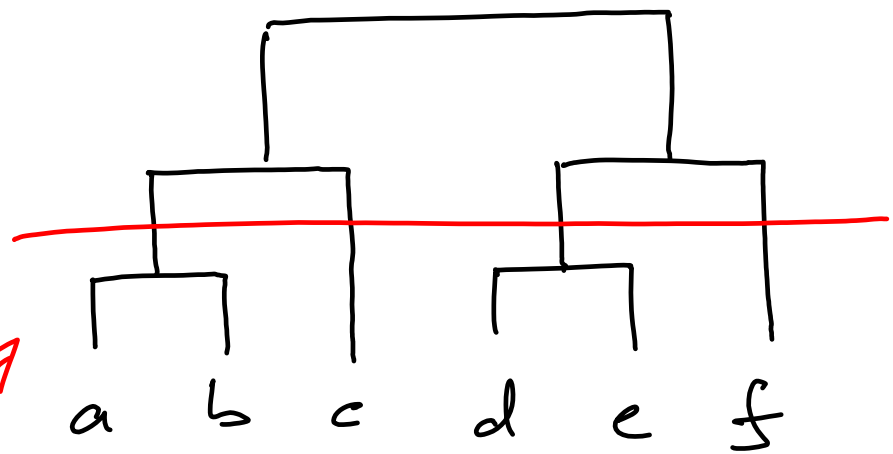
Danas. . .

Dendrogram

Za razliku od particijskog grupiranja, hijerarhijsko grupiranje rezultira hijerarhijom grupa.

Hijerarhija se može prikazati dendrogramom.





presijecanjem dendrograma na željenoj razini dobivamo particijsko grupiranje

Funkcija udaljenosti/mjera sličnosti

Hijerarhijsko grupiranje provodi se na temelju funkcije udaljenosti
$d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$

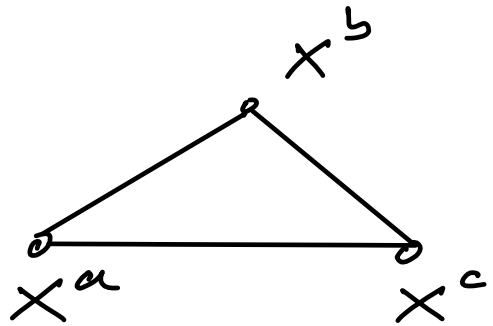
Funkcija udaljenosti zadovoljava svojstva metrike:

(1) $d(\mathbf{x}^a, \mathbf{x}^b) \geqslant 0$

(2) $d(\mathbf{x}^a, \mathbf{x}^b) = 0$ akko $\mathbf{x}^a = \mathbf{x}^b$

(3) $d(\mathbf{x}^a, \mathbf{x}^b) = d(\mathbf{x}^b, \mathbf{x}^a)$

(4) $d(\mathbf{x}^a, \mathbf{x}^b) + d(\mathbf{x}^b, \mathbf{x}^c) \geqslant d(\mathbf{x}^a, \mathbf{x}^c)$



Npr. euklidska udaljenost, Minkowskijeva udaljenost, Mahalanobisova udaljenost

Općenitije, može se koristiti mjera sličnosti/različitosti

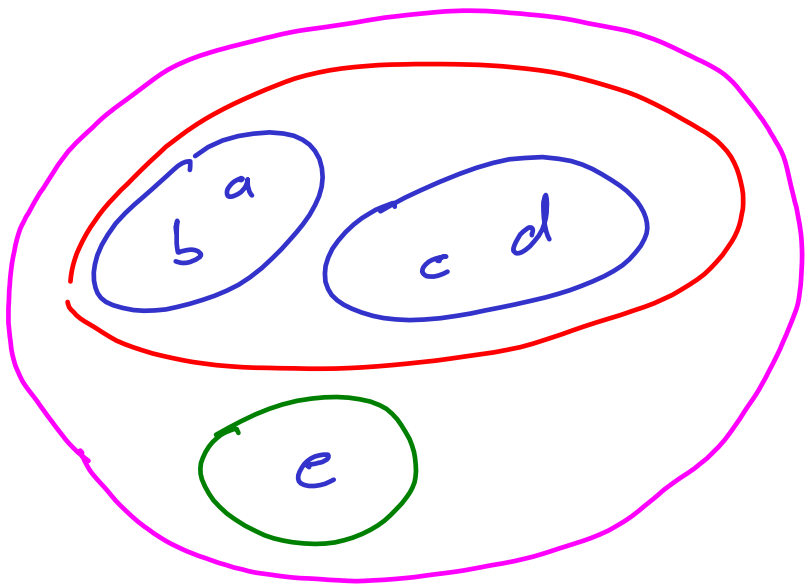
Aglomerativno hijerarhijsko grupiranje

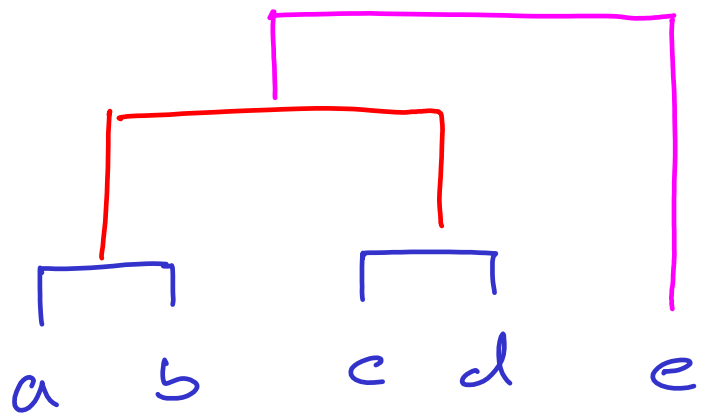
Grupiranje može biti aglomerativno ili divizivno.

Algomerativno grupiranje:

- kreće od grupa koje sadrže svaka po samo jedan primjer
- postepeno stapa najbliže grupe dok sve primjere ne stopi u jednu veliku grupu

Dendrogram se gradi odozdo prema gore (od pojedinačnih primjera do zajedničke grupe).





Stapanje grupa

U svakom koraku treba stopiti dvije najbliže grupe $\mathcal{G}_i$ i $\mathcal{G}_j$. Kako izračunati udaljenost između grupa koje sadrže više od jednog primjera?

- Jednostruko povezivanje (engl. *single linkage*)

$$d_{min}(\mathcal{G}_i, \mathcal{G}_j) = \min_{\mathbf{x} \in \mathcal{G}_i, \mathbf{x}' \in \mathcal{G}_j} d(\mathbf{x}, \mathbf{x}')$$

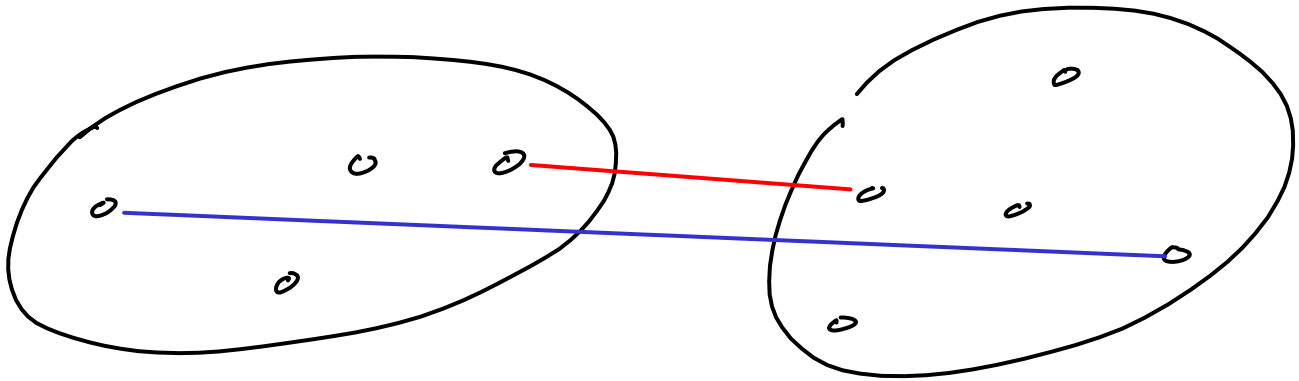
- Potpuno povezivanje (engl. *complete linkage*)

$$d_{max}(\mathcal{G}_i, \mathcal{G}_j) = \max_{\mathbf{x} \in \mathcal{G}_i, \mathbf{x}' \in \mathcal{G}_j} d(\mathbf{x}, \mathbf{x}')$$

- Prosječno povezivanje (engl. *average linkage*)

$$d_{avg}(\mathcal{G}_i, \mathcal{G}_j) = \frac{1}{N_i N_j} \sum_{\mathbf{x} \in \mathcal{G}_i} \sum_{\mathbf{x}' \in \mathcal{G}_j} d(\mathbf{x}, \mathbf{x}')$$

Stapanje grupa



Jednostruko povezivanje

Potpuno povezivanje

Algoritam HAC

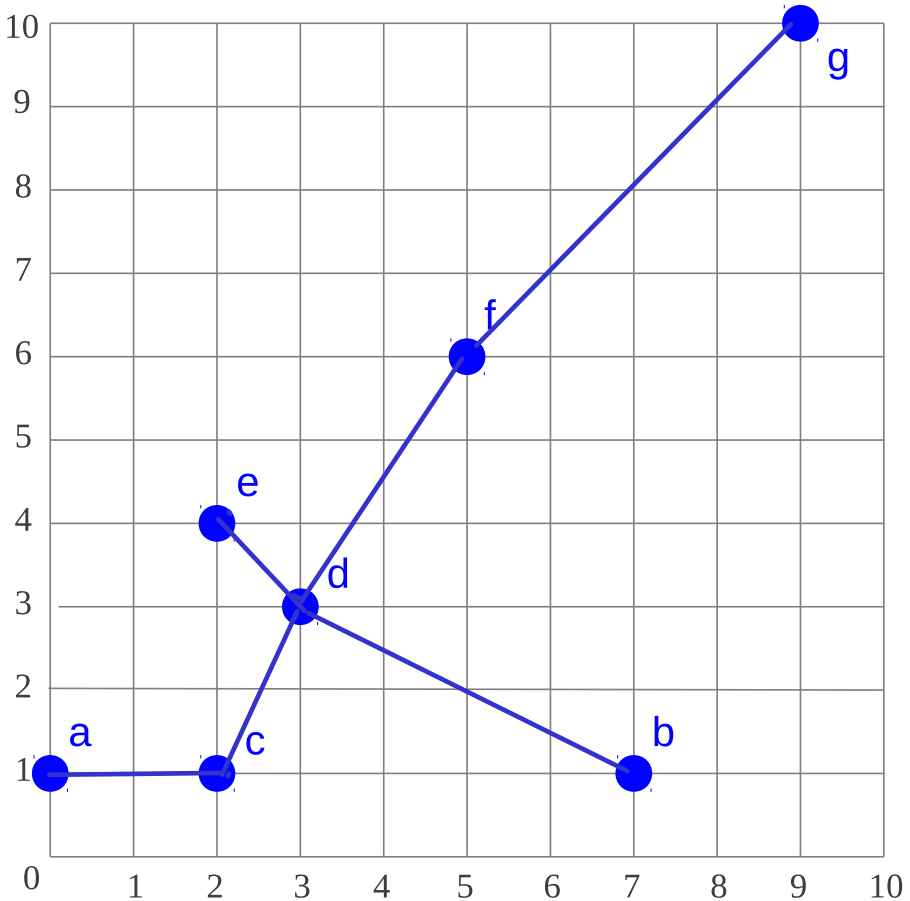
Algoritam hijerarhijskog aglomerativnog grupiranja (HAC)

inicijaliziraj K, $k \leftarrow N$, $\mathcal{G}_i \leftarrow \{\mathbf{x}^{(i)}\}$ za $i = 1, \ldots, N$
ponavljaj
  $k \leftarrow k - 1$
  $(\mathcal{G}_i, \mathcal{G}_j) \leftarrow \underset{\mathcal{G}_a, \mathcal{G}_b}{\operatorname{argmin}} d(\mathcal{G}_a, \mathcal{G}_b)$
  $\mathcal{G}_i \leftarrow \mathcal{G}_i \cup \mathcal{G}_j$
dok je $k > K$

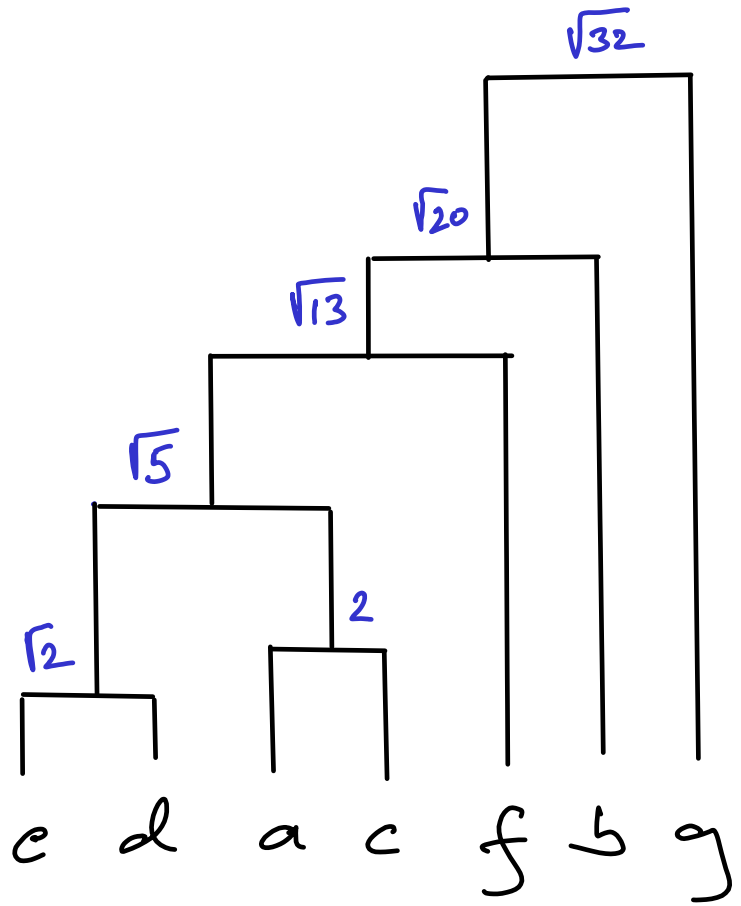
K je unaprijed zadani broj grupa

Za $K = 1$ dobivamo potpun dendrogram koji možemo naknadno presijecati

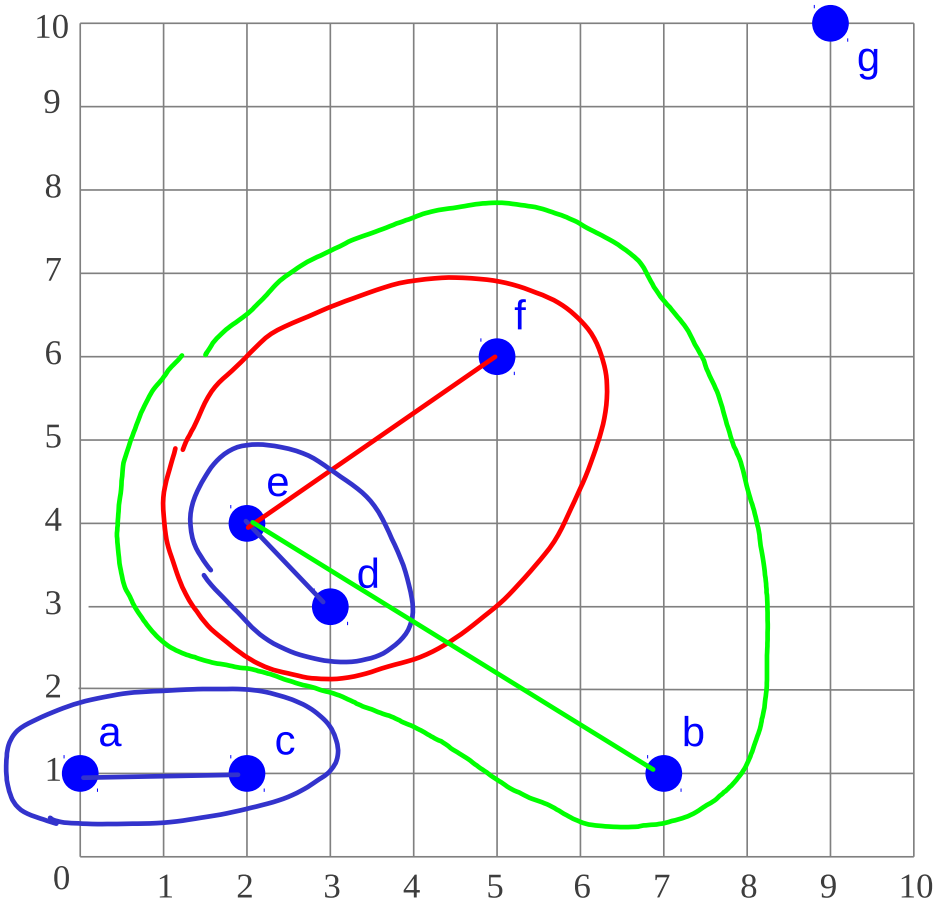
Algoritam HAC – primjer 1



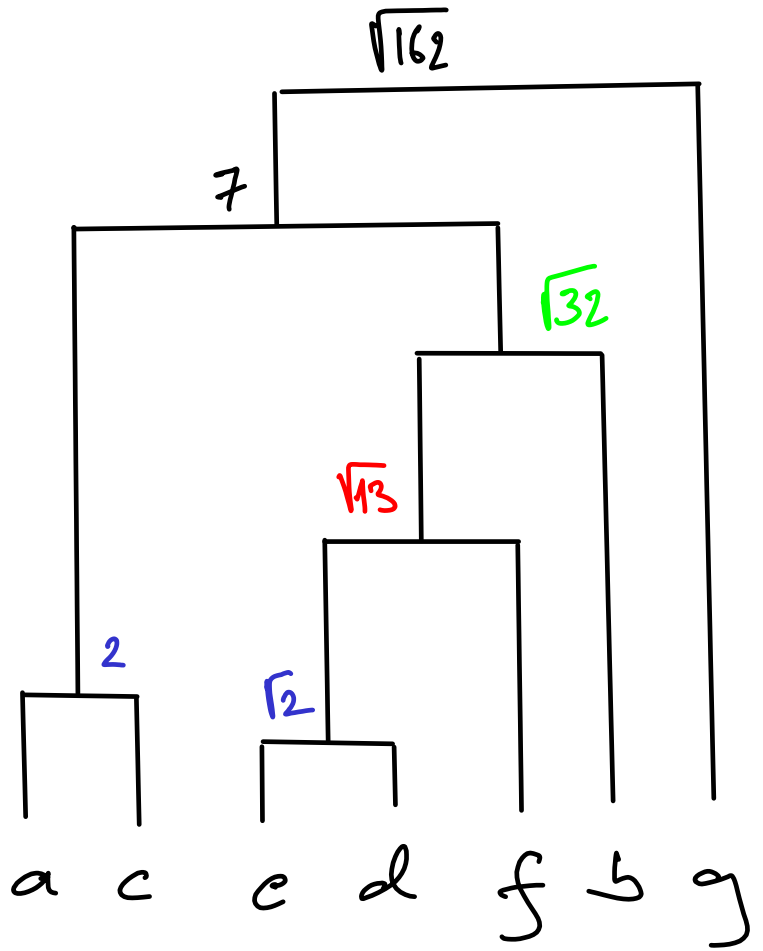
Jednostruko povezivanje:



Algoritam HAC – primjer 2



Potpuno povezivanje:



Algoritam HAC – složenost

Prostorna složenost:

matrica udaljenosti sadržava udaljenosti između $\binom{N}{2}$ parova primjera

$\Rightarrow \mathcal{O}(N^2)$ ← problematično!

Vremenska složenost:

broj koraka algoritma: $N - K$

pronalaženje najbližeg para grupa: $\mathcal{O}(N^2)$ ← vrlo problematično!

$\Rightarrow \mathcal{O}((N-K)N^2) \approx \mathcal{O}(N^3)$

$K \ll N$

Implementacija s prioritetnom listom: $\mathcal{O}(N^2 \log N)$

Jednostruko povezivanje: $\mathcal{O}(N^2)$

Danas. . .

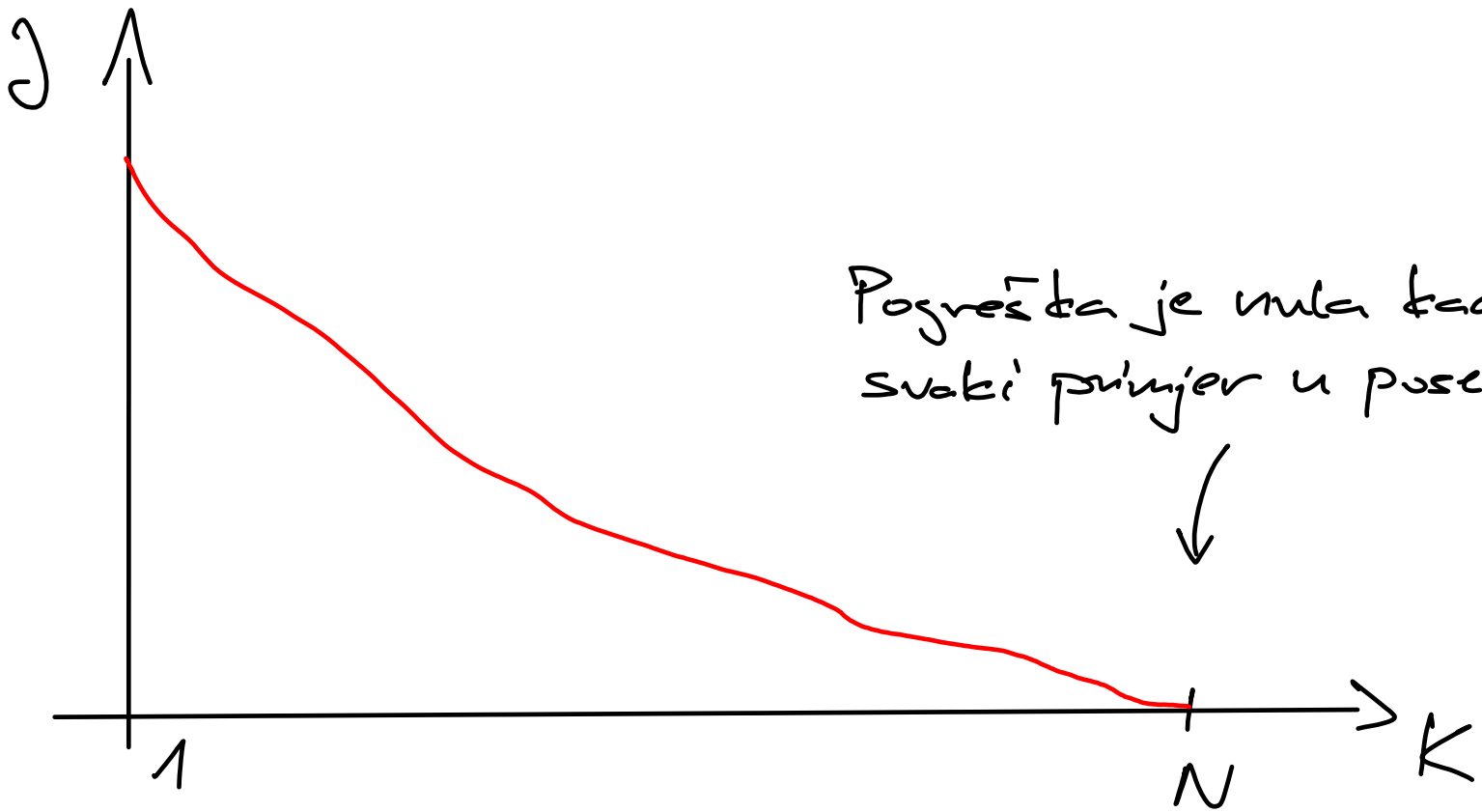
Broj grupa K

Kod svih algoritama broj grupa K treba unaprijed odrediti!

Kako odrediti optimalan K?

Odabrati K koji minimizira pogrešku J?

$$J = \sum_{k=1}^{K} \sum_{i=1}^{N} b_k^i \|x^i - \mu_k\|^2$$



Pogreška je nula kada je svaki primjer u posebnoj grupi

Provjera grupa

Problem odabira broja grupa analogan odabiru složenosti modela kod nadziranog učenja.

Dio većeg problema koji se zove provjera grupa (engl. *cluster validation*): *koliko je dobro naše grupiranje?*

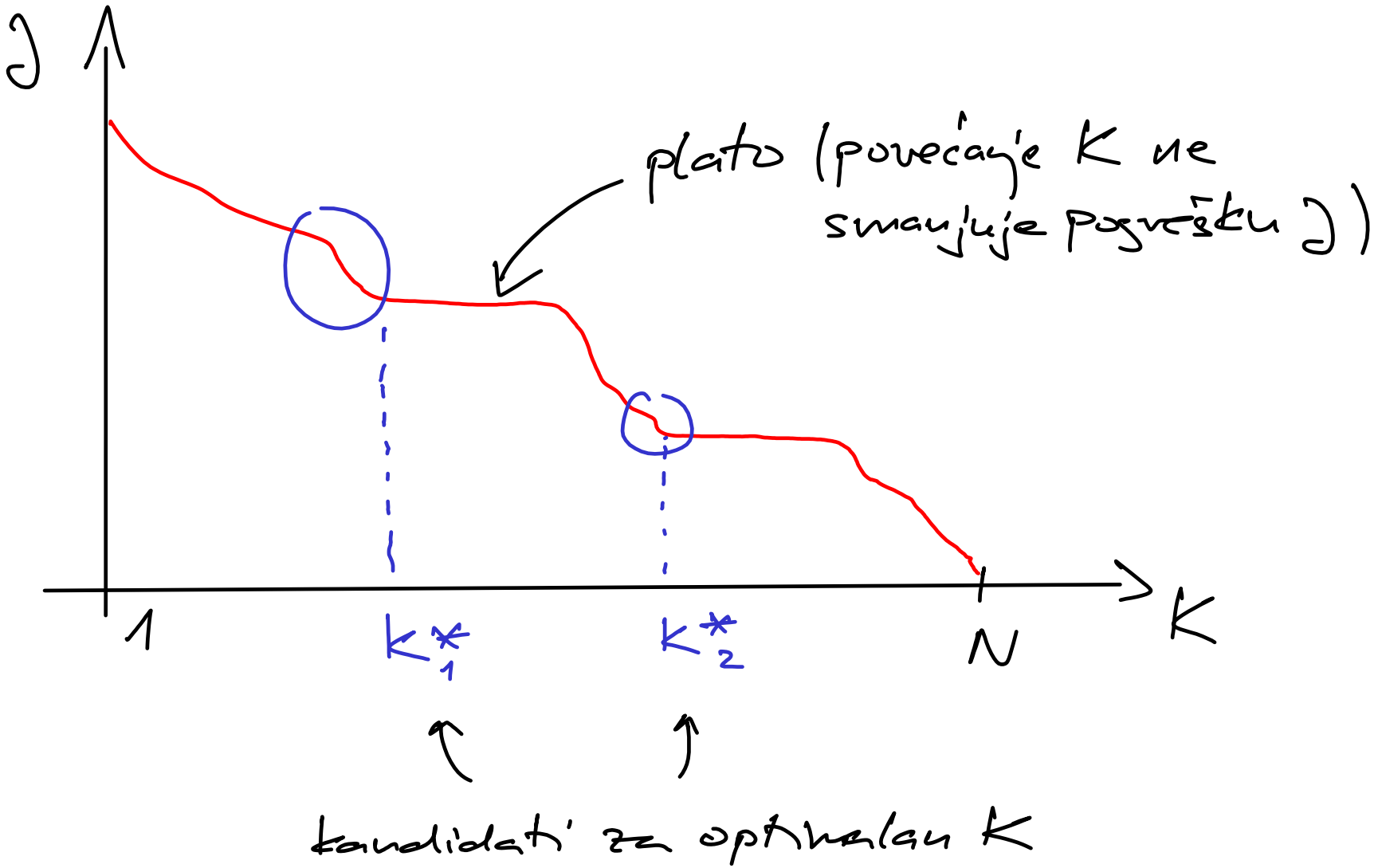
Neke metode:

- ručna provjera kvalitete grupa
- preslikavanje u 2D-prostor i vizualna provjera
- metoda "koljena"
- provjera na označenom podskupu primjera (npr. Randov indeks)
- minimizacija regularizirane funkcije pogreške

$$K^* = \operatorname*{argmin}_K \big(J(K) + \lambda K\big)$$

kompromis između smanjenja J i povećanja K

Metoda "koljena"



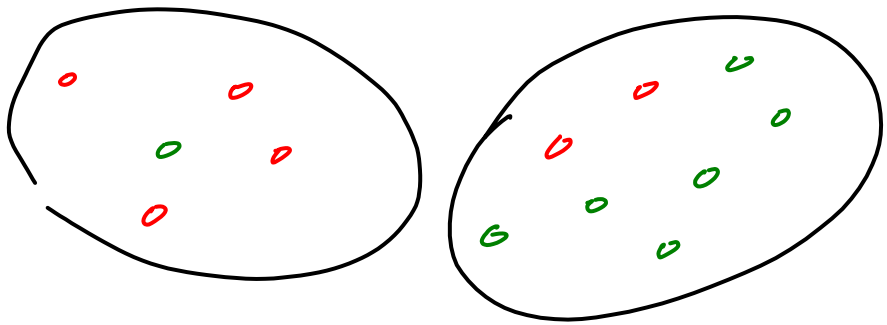
Randov indeks

$$R = \frac{a+b}{\binom{N}{2}}$$

← Odabiremo K koji maksimizira R

a – broj jednako označenih parova u istim grupama

b – broj različito označenih parova u različitim grupama



$N = \binom{13}{2} = 78$

$a = \binom{4}{2} + \binom{2}{2} + \binom{1}{2} + \binom{6}{2} = 22$ (=6, =1, =0, =15)

$b = 4 \cdot 6 + 1 \cdot 2 = 26$

$R = \frac{22+26}{78} = 0,62$

Sažetak

- Ako primjeri nisu označeni (ne znamo ih označiti ili je preskupo) moramo koristiti **nenadzirano strojno učenje**
- Tipičan zadatak je **grupiranje**, kojim se primjeri razdjeljuju u grupe prema udaljenosti/sličnosti
- Grupiranje može biti **particijsko** ili **hijerarhijsko**
- **Algoritam k-srednjih vrijednosti** računa središta grupa i primjere razdjeljuje u grupe s najbližim središtima, smanjujući postepeno funkciju pogreške. **Odabir početnih središta** utječe na grupiranje
- **Hijerarhijsko algomerativno grupiranje** postepeno stapa najbliže grupe temeljem **funkcije udaljenosti** i gradi **dendrogram**
- Ključan problem grupiranja jest **odabir optimalnog broja grupa**

Sljedeća tema: Algoritam maksimizacije očekivanja