Gépi tanulás a gyakorlatban

Kiértékelés és Klaszterezés

Kiértékelés

- Hogyan alkalmazzuk sikeresen a gépi tanuló módszereket?
  - Hogyan válasszuk az algoritmusokat?
  - Hogyan hangoljuk a paramétereiket?
- Precízebben:
  - Tegyük fel, hogy tanítunk egy Logisztikus regressziót és azt tapasztaljuk, hogy a tanítás után elfogadhatatlanul nagy a hiba. Mit tegyünk?
  - Opciók:
    - Nagyobb tanító adatbázist szerzünk. Ez mindig segít?
    - Kisebb jellemzőkészlet használata
    - Újabb (várhatóan reprezentatívabb) jellemzők keresése
    - Új jellemzők bevezetése a meglévők alapján (pl. RBF-ek vagy kernelek használata)
    - Regularizációs paraméter állítása

Kiértékelés

- Kiértékelés:
  - Ahhoz, hogy leteszteljük egy tanítást szükség van egy *kiértékelő halmazra* (teszt halmazra, evaluation set) és egy *kiértékelési metrikára*
  - Ennek függetlennek kell lennie a tanító halmaztól!
  - Viszont karakterisztikájában meg kell egyeznie azzal! → pl. hasonló osztály címke eloszlás
  - Tipikus eset:
    - Kiértékelő adatbázis: Teljes adatbázis két részre bontása:
      - 80% tanító adatbázis → tanítás során használjuk
      - 20% kiértékelő adatbázis → kiértékelésre használjuk
    - Kiértékelési metrika: Használjuk az algoritmusok által minimalizált költség függvény értékét

Kiértékelés

- További metrikák:
  - Átlagos 0-1 hiba: err(h(x),y) = 1, ha tévesztés történt 0 különben. Ezek átlagolása az adatbázison. Kiegyensúlyozatlan eset? Másként: (FP+FN)/(TP+FP+TN+FN)
  - Tévesztési mátrix (két osztályra):
    - Fedés (recall): Pozitívak mekkora részét találtuk el: TP/(TP+FN)
    - Pontosság (precision): A pozitív predikciók mekkora része helyes: TP/(TP+FP)
    - F1-mérték: a fenti kettő harmonikus közepe → Jó kiegyensúlyozatlan címke eloszlás esetén

		Prediction	
		0	1
Actual	0	TN	FP
	1	FN	TP

Kiértékelés

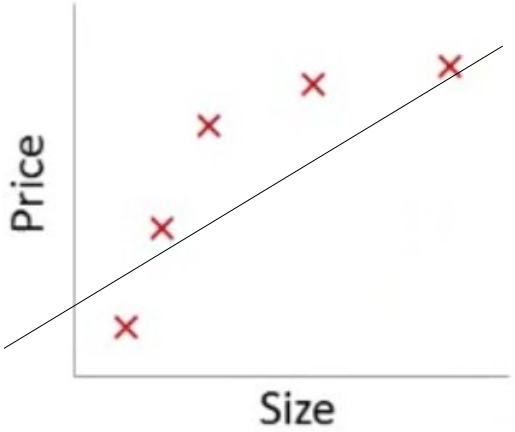
- Ahhoz, hogy leteszteljük egy tanítást szükség van egy *kiértékelő halmazra* (teszt halmazra, evaluation set) és egy *kiértékelési metrikára*
  - Legyen adott egy tanító és kiértékelő részre szétvágott adatbázis
  - Tanítsunk *több* polinomiális regressziót, különféle fokszám paraméterek mellett a tanító halmazon, majd nézzük meg melyik a legjobb a kiértékelő halmazon:
    - D= 1 → h1_Theta1 → J_Test(Theta1)
    - D= 2 → h2_Theta2 → J_Test(Theta2)
    - D= 3 → h3_Theta3 → J_Test(Theta3)
    - D= 4 → h4_Theta4 → J_Test(Theta4)
  - Azt tapasztaljuk, hogy D=2 fokszám választás adja a legjobb eredményt. Így ezzel szállítjuk a modellt a megrendelőnek, aki – némi használat után – elégedetlenségét fejezi ki (rossz a modell). Mi történt?

Kiértékelés

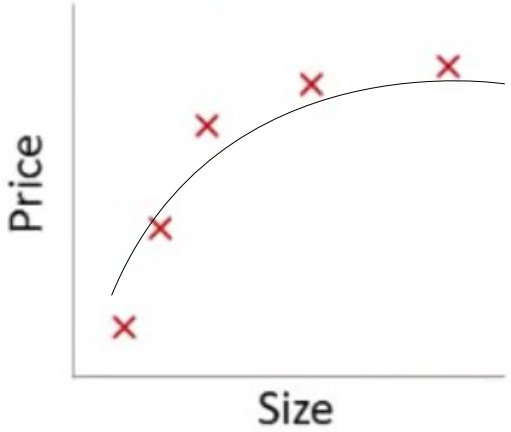
- Válasz: A kiértékelő adatbázist is tanító adatbázisként használtuk a *d* (fokszám) tanulása során!
- Megoldás: Validációs halmaz bevezetése
  - Tanító adatbázis 60%, Validációs adatbázis 20%, Kiértékelő adatbázis 20%
  - A kiértékelő adatbázisnak *függetlennek* kell lennie mindenféle tanítástól, paraméter hangolástól!
  - Ennek igaznak kell lennie az emberi tényezőre is! → N-Fold Cross Validation → WEKA támogatás!
- Helyes folyamat:
  - D= 1 → h1_Theta1 → J_Validation(Theta1)
  - D= 2 → h2_Theta2 → J_Validation(Theta2)
  - D= 3 → h3_Theta3 → J_Validation(Theta3)

  … → Megfelelő D választása → Teszt hiba számítása

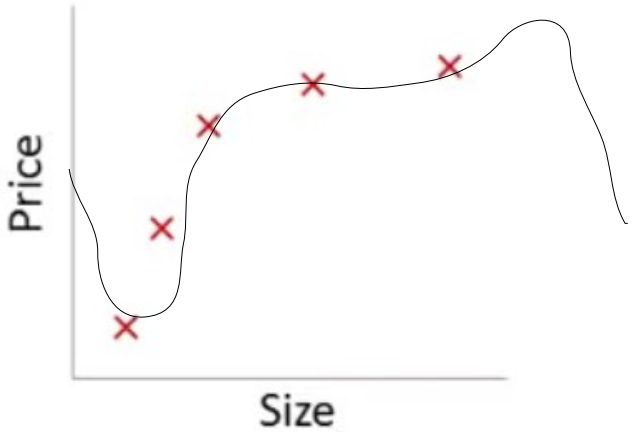
Kiértékelés



$\theta_0 + \theta_1 x$



$\theta_0 + \theta_1 x + \theta_2 x^2$

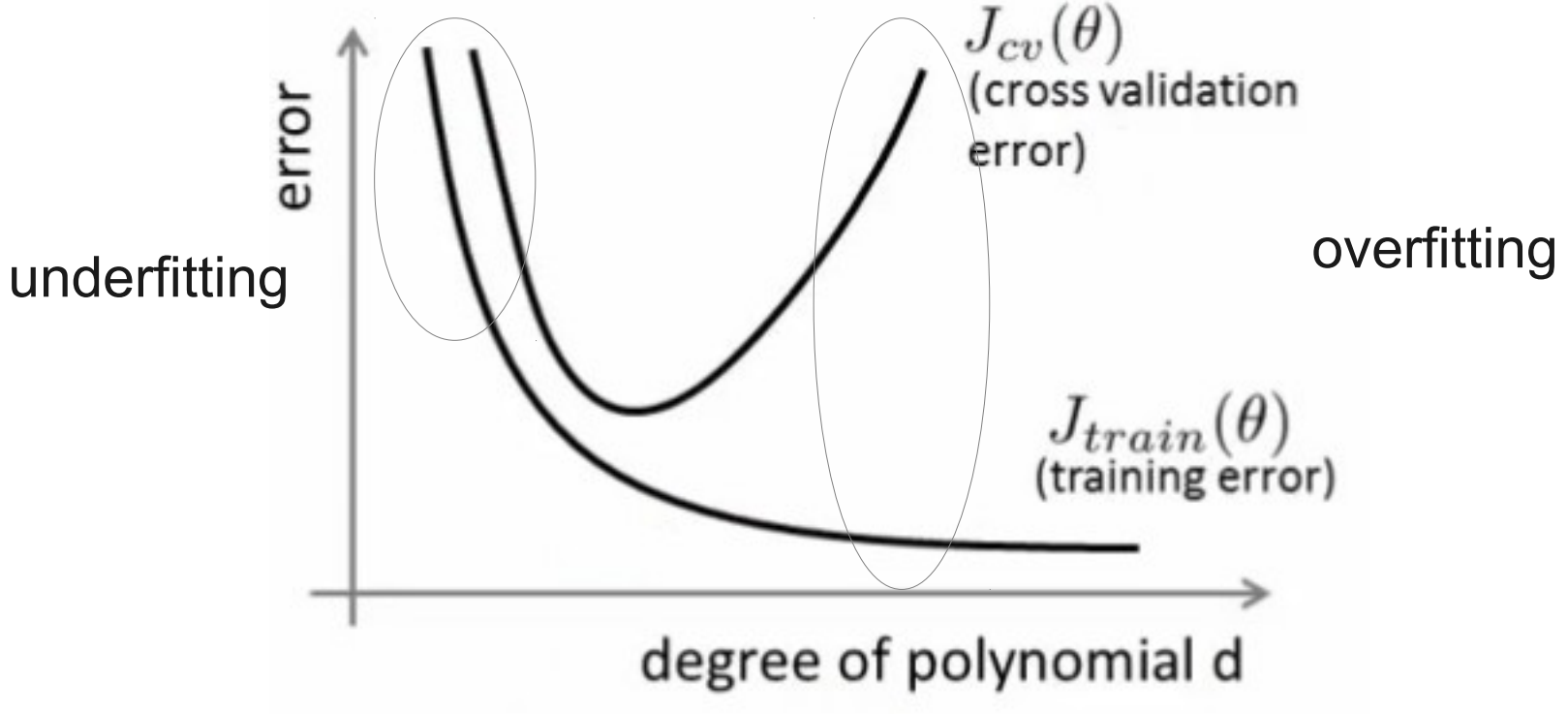


$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

- Nagy a teszt hiba. Mi lehet az oka. Tipikus esetek:
  - Alultanulás (high bias problem, underfitting) → a modell nem képes reprezentálni az adatpontokat, nem elég rugalmas az adott tanulási feladathoz
  - Túltanulás (high variance problem, overfitting) → a modell túl rugalmas, túlreprezentálja a tanuló halmazt, azaz a tanuló pontok sajátosságait is képes elkapni, a leírt sokaság általános tulajdonságain túl

Kiértékelés

- Vizsgáljuk meg a korábban említett polinomiális regresszió fokszáma függvényében a tanuló és validációs hiba alakulását.



- Automatikus paraméter hangolásra ad lehetőséget.
- Mi az oka annak, ha ennek ellenére J_test != J_validation?

Kvíz

Consider regularized logistic regression. Let

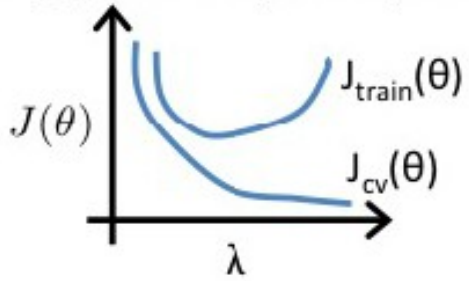
$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

$$J_{\text{train}}(\theta) = \frac{1}{2m_{\text{train}}}\left[\sum_{i=1}^{m_{\text{train}}}(h_\theta(x_{\text{train}}^{(i)}) - y_{\text{train}}^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

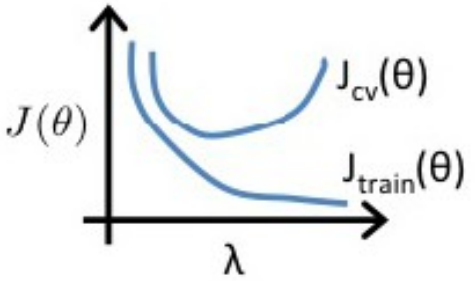
$$J_{\text{cv}}(\theta) = \frac{1}{2m_{\text{cv}}}\left[\sum_{i=1}^{m_{\text{cv}}}(h_\theta(x_{\text{cv}}^{(i)}) - y_{\text{cv}}^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

Suppose you plot J_{train} and J_{cv} as a function of the regularization parameter λ. Which of the following plots do you expect to get?

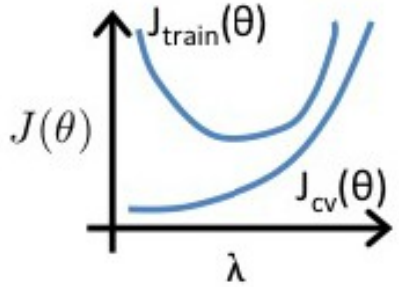
1.



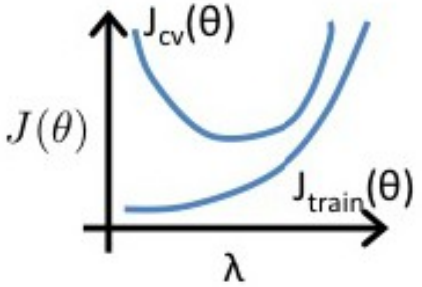
2.



3.

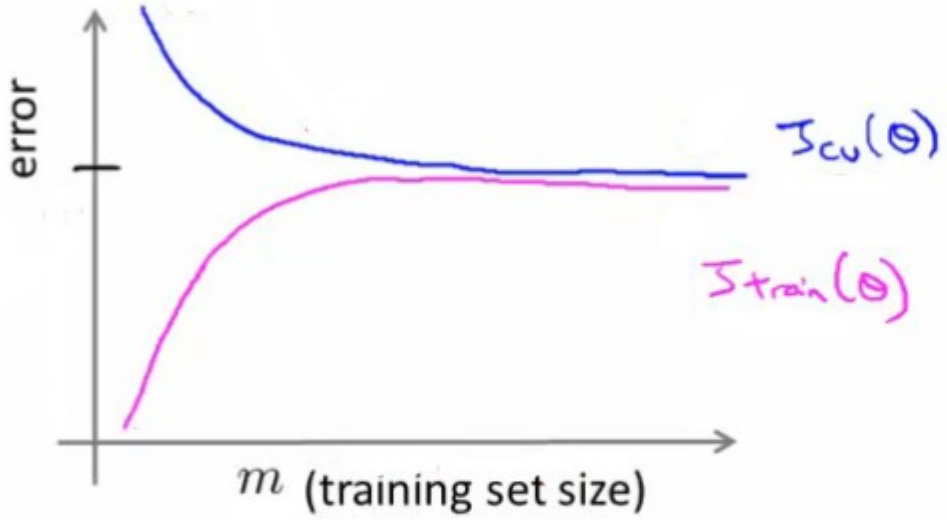


4.



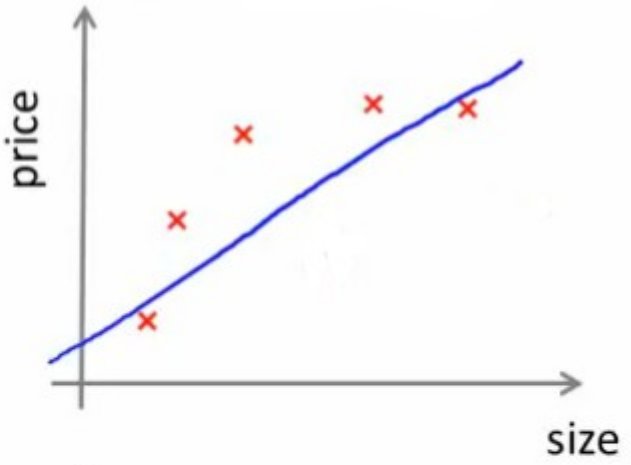
Kiértékelés

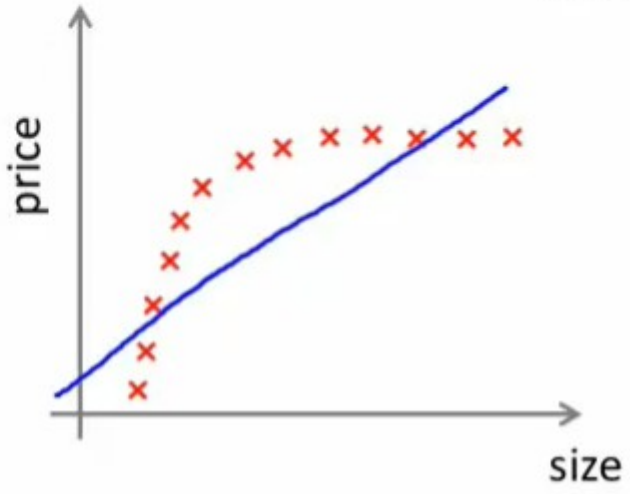
High bias



- Hiba magas
- Több példa nem segít

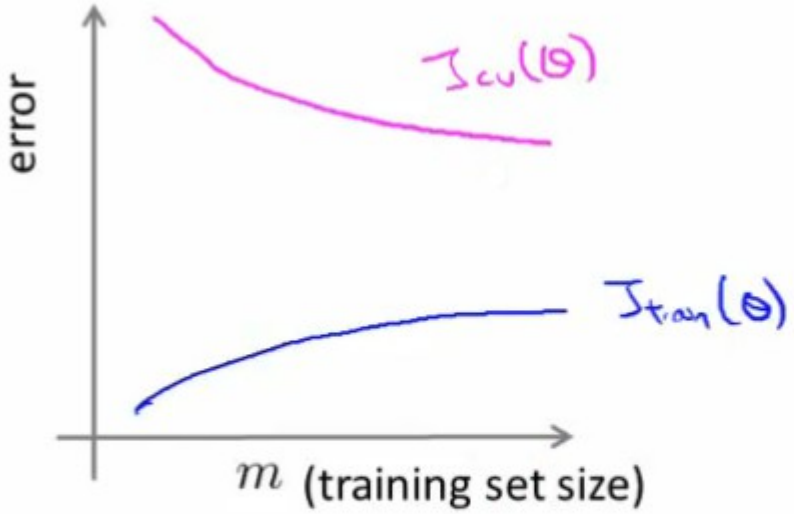
$h_\theta(x) = \theta_0 + \theta_1 x$



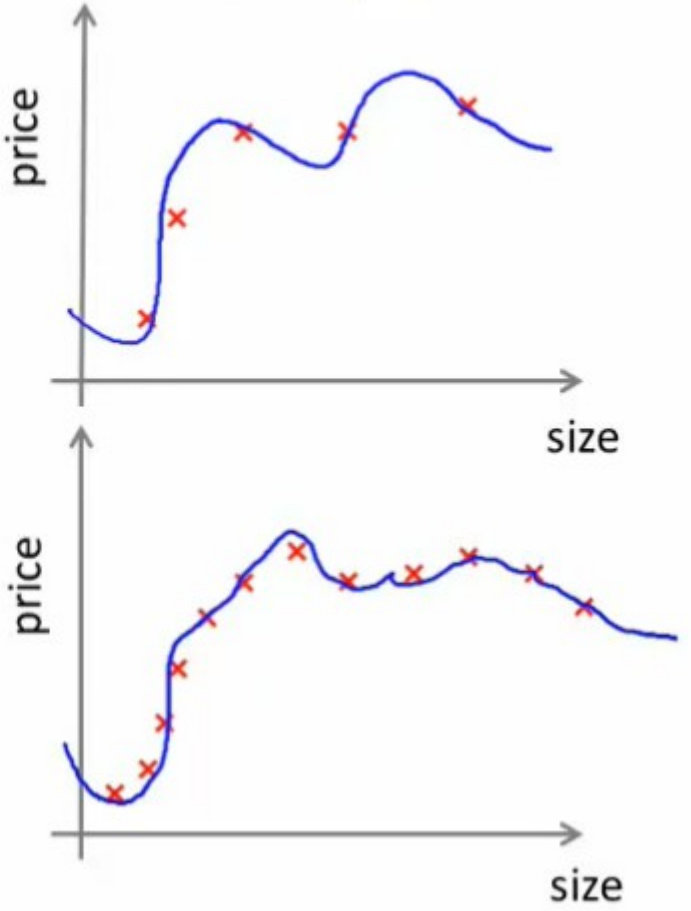


Kiértékelés

High variance



$h_\theta(x) = \theta_0 + \theta_1 x + \cdots + \theta_{100} x^{100}$
(and small λ)



- Hézag a validációs és a tanító halmaz költsége között
- Több példa segít

Kiértékelés

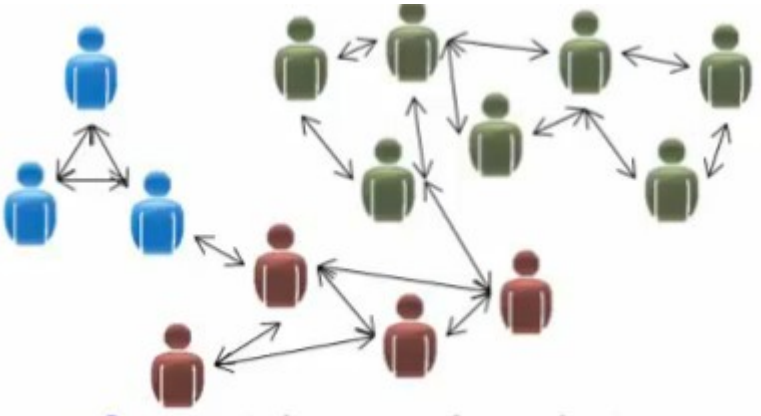
- Összefoglalás, melyik hiba típus esetén mi segít:
  - Nagyobb tanító adatbázist szerzünk. → Túltanuláson segít, mivel összébb húzza a tanító halmaz mérete függvényében kirajzolt költségeket (előző slide)
  - Kisebb jellemzőkészlet használata → Túltanuláson segít, hiszen „sűríti” a teret, a dimenzió csökkentése által
  - Újabb (várhatóan reprezentatívabb) jellemzők keresése → Jobban tanulhatóvá teszi a problémát
  - Új jellemzők bevezetése a meglévők alapján (pl. RBF-ek vagy kernelek használata) → Alultanuláson segít, mivel lehetővé teszi, hogy bonyolultabb döntési felületet tanuljunk egyszerű modellel.
  - Regularizációs paraméter állítása:
    - Csökkentés → Segíti az alultanulás elkerülését
    - Növelés → Segíti a túlillesztés elkerülését

Kiértékelés

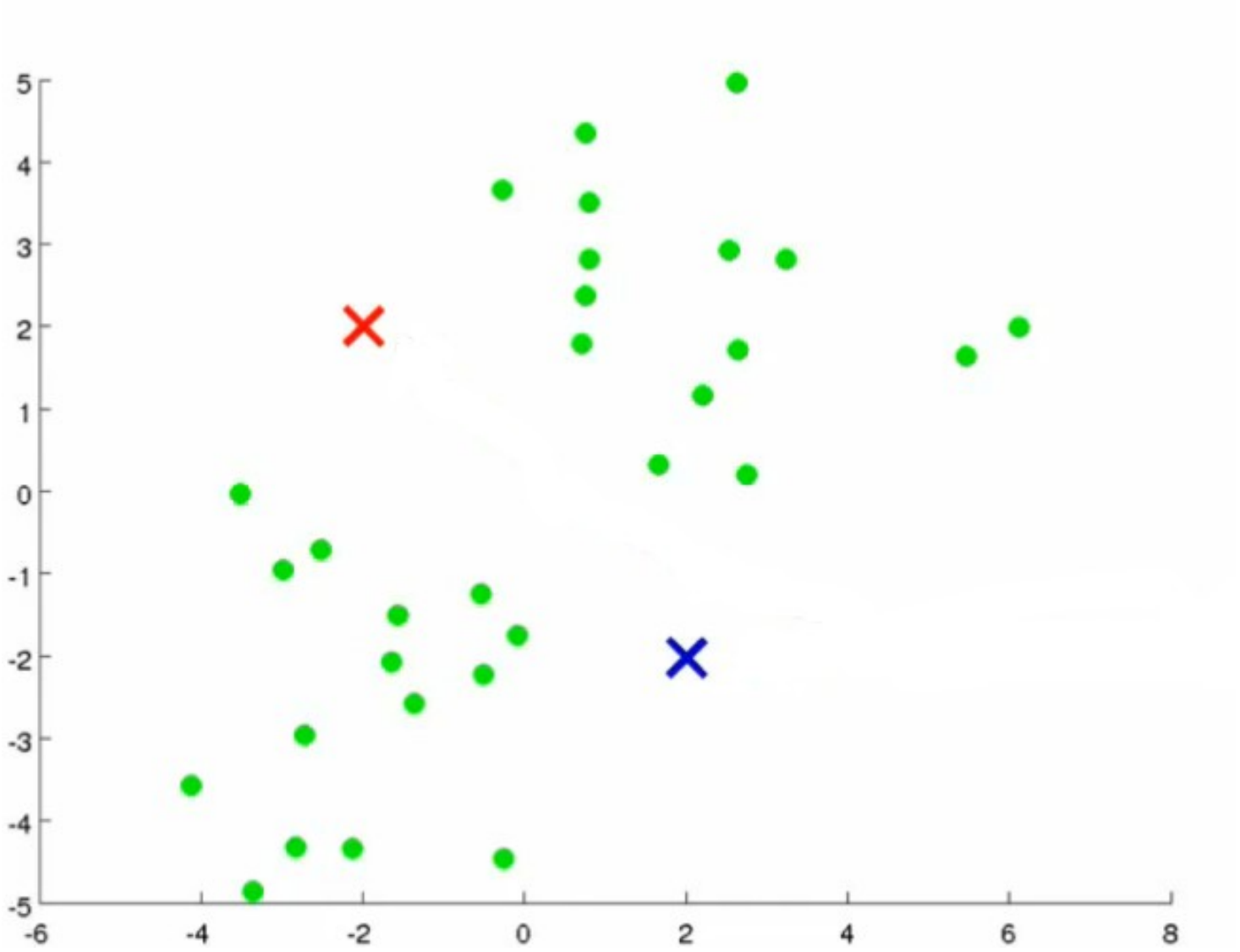
- Gépi tanulási algoritmusok alkalmazása:
  - Alkalmazzuk a legegyszerűbb algoritmusokat
  - Vizsgáljuk a paraméterek és a tanuló halmaz méretének függvényében a költség (hiba) alakulását a validációs halmazon
  - Indokoljuk meg a hibát → célirányosan javítjuk
  - Figyeljünk arra, hogy a teszthalmaz független legyen → Manuálisan se használjuk paraméter hangolásra → Használjunk 10-Fold Cross Validation-t tényleges randommal, hogy megbízhatóbb eredményeket kapjunk.
  - A jól tanulható, már ismert adathalmazon tanítsuk a végleges modellt

Klaszterezés

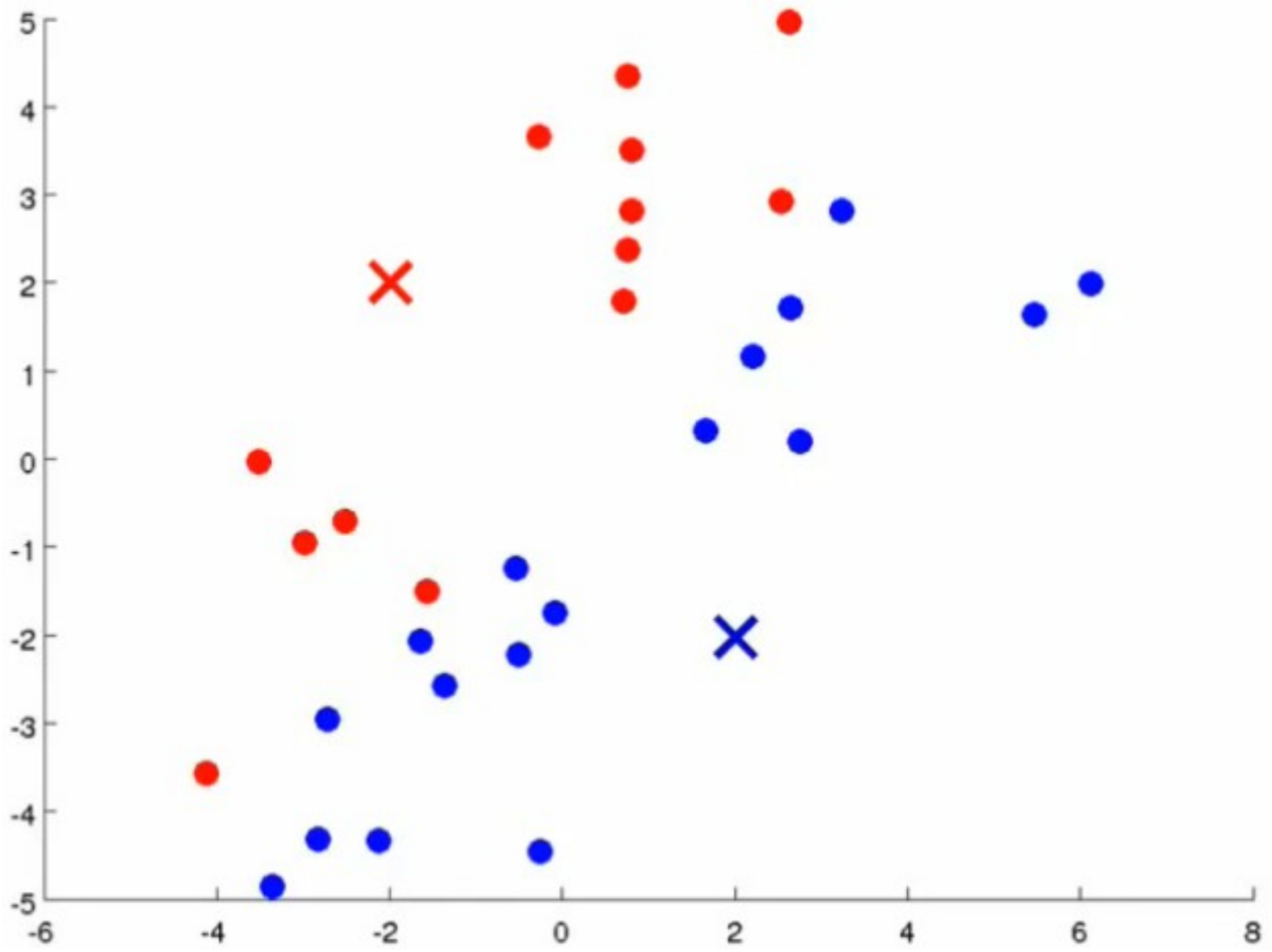
- Felügyelet nélküli módszer
- Csak a tanító adatpontok adottak, nincs segéd információ (címkék)
- Cél: az adatbázis belső struktúrájának feltérképezése segítségével tanulni
  - Klaszterezés esetén:
    - Csoportok detektációja: az egymáshoz közeli egyedek kerüljenek egy csoportba, mialatt a klaszterek (csoporotok) legyenek egymástól a lehető legtávolabb
    - Alkalmazási példák:



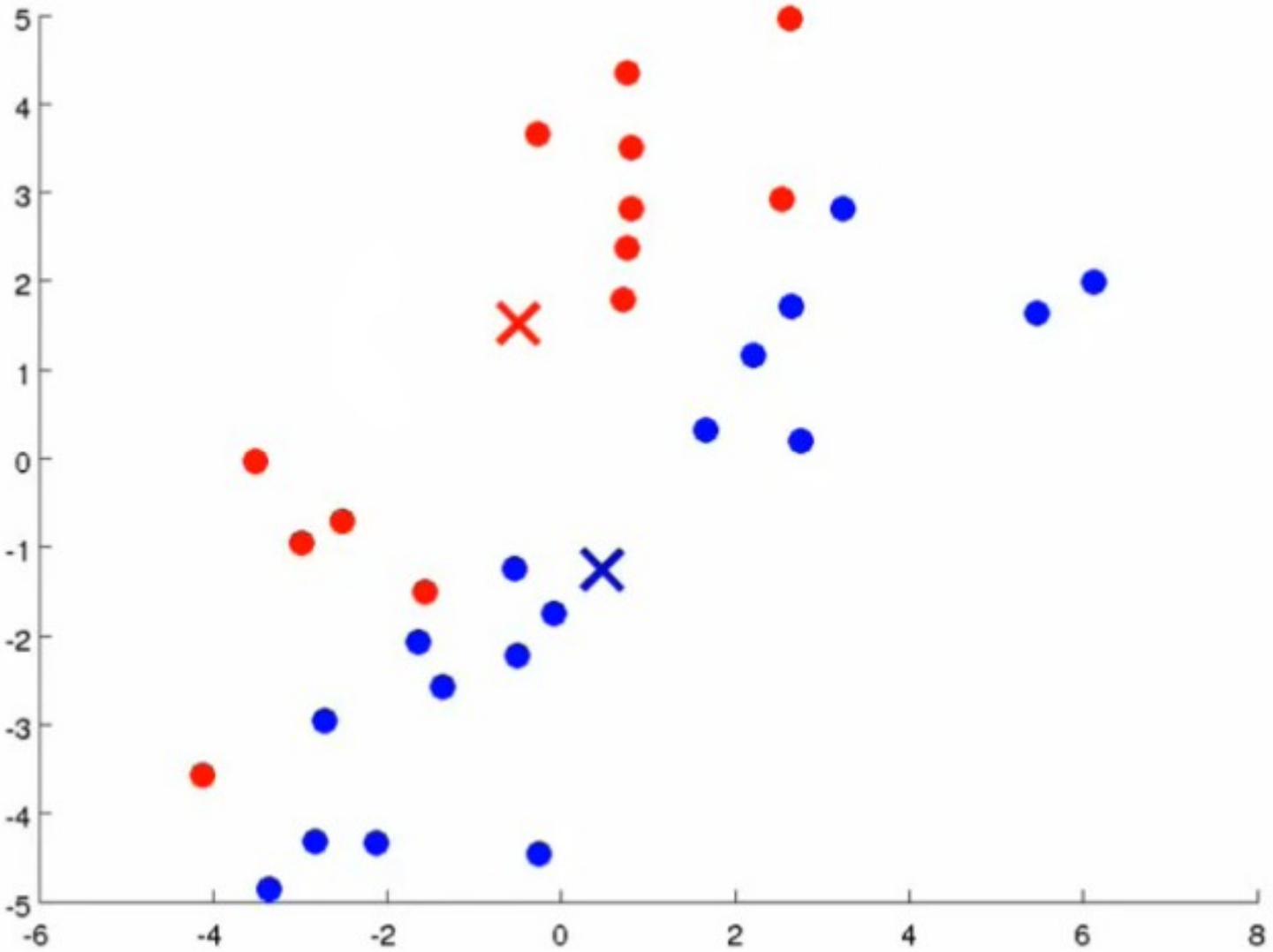
Klaszterezés - K-Means



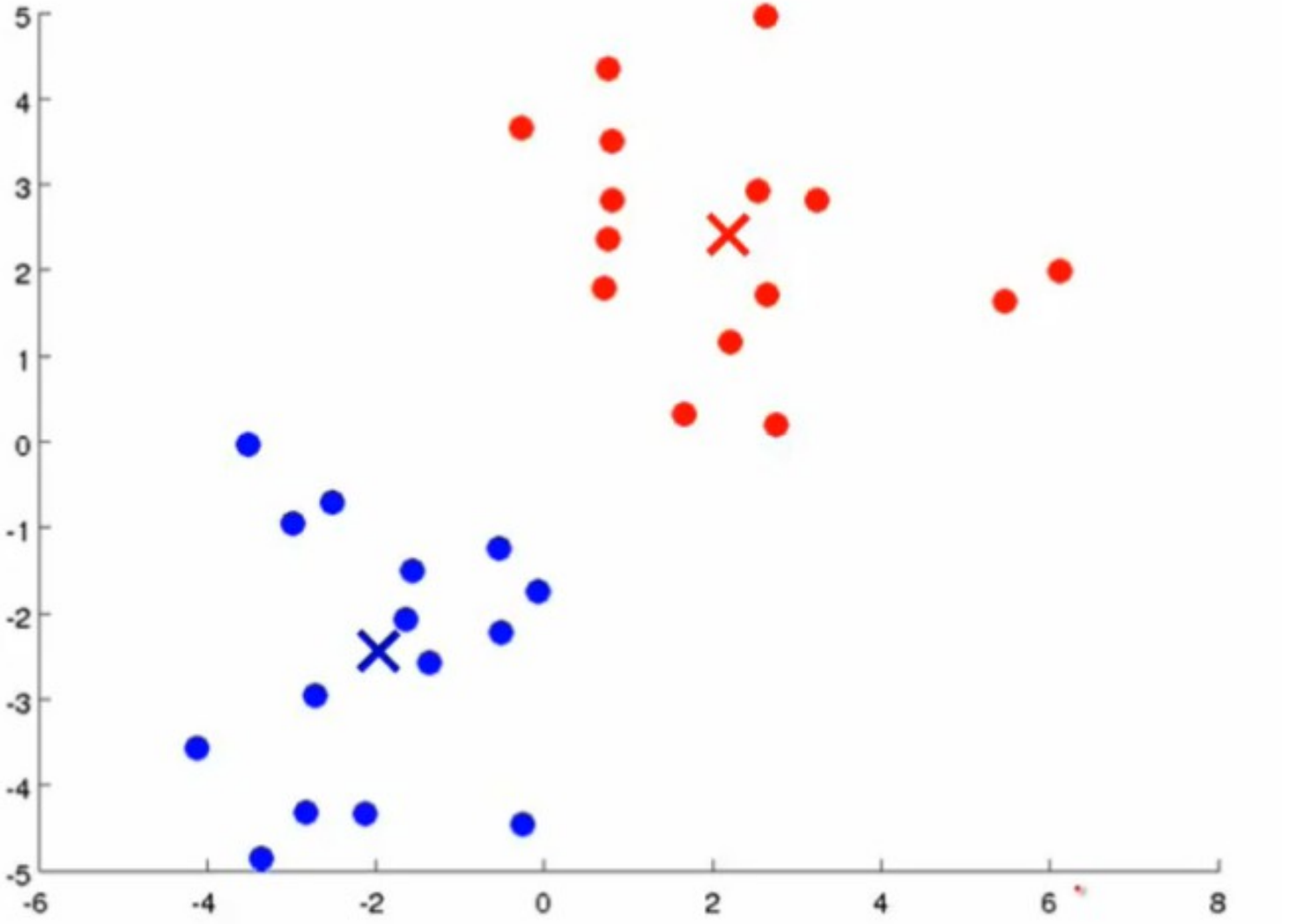
Klaszterezés - K-Means



Klaszterezés - K-Means



Klaszterezés - K-Means



Klaszterezés - K-Means

- K-Means is megfogalmazható J költségfüggvény minimalizálásos alakban:

Jelölések:

μ_i: az i-edik klaszter középpontja.

$\mathbf{r_{ij}} = 1$, ha az i-edik klaszterbe soroljuk a j-edik mintát; 0, egyébként.

Ezek alapján fölírhatunk egy hibafüggvényt:

$$Err = \sum_i \sum_j r_{ij} ||\mu_i - x_j||^2 \rightarrow min$$

Algoritmus:

1. inicializáljuk a μ_j klaszzerközéppontokat (pl. egyenletes eloszlással a térben)
2. legyen $r_{ij} = 1$ ha az x_i a μ_j-hez van legközelebb, egyébként 0. (minták középponthoz rendelése)
3. legyen $\mu_j = \frac{\sum_i r_{ij} x_i}{\sum_i r_{ij}}$ minden $j = 1, 2, \ldots, k$-ra. (középpontok újradefiniálása)
4. ismételjük a 2. ponttól, amíg változik valamelyik klaszterközéppont.

Problémák a $k-means$-el:

1. ha egy csoportra két közepet inicializálunk (túl nagy k választása)
2. ha két csoport közé inicializálunk egy közepet (túl kicsi k választása)

Klaszterezés

- Az algoritmusnak van két változata:
  - EM klaszterező:
    - A szigorú klaszterközépponthoz rendelés helyett → klaszterhez tartozási valószínűségek számítása
    - Új középpont számítása minden pontra a fenti valószínűségek súlyozása mellett
  - X-Means:
    - Heurisztikus döntés arról, hogy egy klaszter mennyire diverz → küszöb fölött kettévág új klaszterközéppontok bevezetésével
- K-Means esetén kezdeti középpontok és azok számának meghatározása:
  - Többször futtatva → megbízhatóbb eredmények
  - K mint paraméter hangolása → Költség számítása K függvényében → „Könyök” keresése

Klaszterezés – Aglomeratív klaszterezők

1. minden példa legyen egy klaszter
2. legyen C a klaszterek halmaza
3. legyen i, j (a C i-edik, és j-edik eleme), amelyre $d_{ij} \rightarrow min$
4. legyen k az i, j elemek apja
5. legyen $C = C \cup k$
6. definájuk k távolságát a többi elemtől C-ben
7. $C = C/\{i, j\}$
8. ismételjük a 3-tól, amíg $|C| \neq 1$.

Legyen i és j egy-egy klaszter, amelyeket összevonunk egy $[ij]$ klaszterré. Ekkor az $[ij]$ klaszter távolsága a többi k klasztertől a következő képpen számolható ki négy eltérő módszer alapján:

1. Single-linkage (nearest neighbor):

$$d_{k[ij]} = minimum(d_{ki}, d_{kj})$$

2. Complete-linkage (furthest neighbor):

$$d_{k[ij]} = maximum(d_{ki}, d_{kj})$$

3. Weighted Pair Group Method using Arithmetic averages (WPGMA):

$$d_{k[ij]} = \frac{d_{ki} + d_{kj}}{2}$$

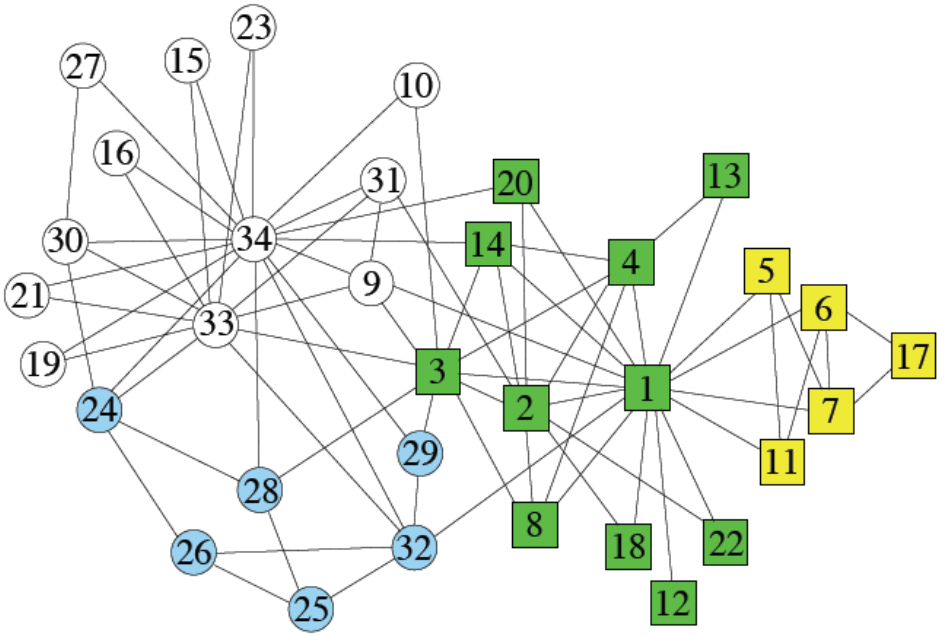
4. Unweighted Pair Group Method using Arithmetic averages (UPGMA):

$$d_{k[ij]} = \frac{n_i}{n_i + n_j} * d_{ki} + \frac{n_j}{n_i + n_j} * d_{kj},$$

ahol az n_i az i-edik klaszter elemszáma.

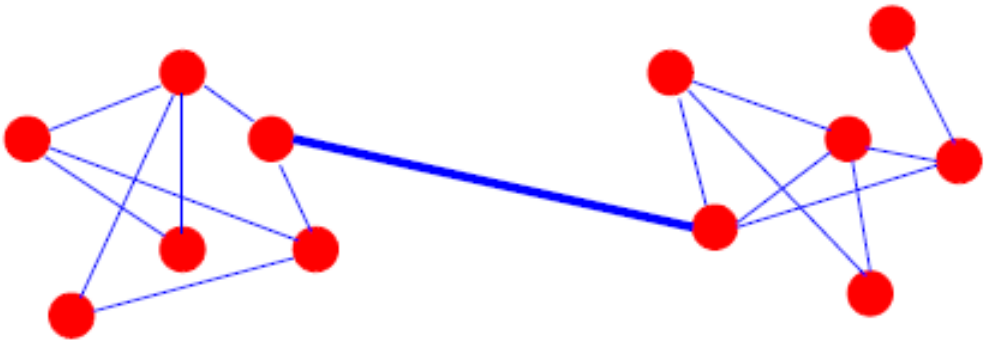
Klaszterezés – Gráf klaszterezők

- Gyakran a bemenet nem az Euklideszi tér elemei, hanem összetettebb struktúrák (pl. gráfok)
  - Ezek klaszterezésére speciális algoritmusok léteznek
  - Célkitűzés:
    - Valamilyen objektumok halmaza felett minták, csoportok detektálása csakis az egyedek közötti kapcsolatok struktúrájának a felhasználásával, valamint esetleges hierarchikus szerveződések feltárása
    - Klaszterek lehetnek átfedőek (Community)



Klaszterezés – Gráf klaszterezők

- Girvan Newman algoritmus:
  - Él központiság számítása: bármely két pont pár között a legrövidebb utak hányszor érintik ugyan azt az élet
  - A „legközpontibb” él mentén kell vágni a gráfot
- Módosítások:
  - Random-walk alapú megközelítés: nem minden pont között legrövidebb utak, hanem véletlen séták mentén
  - Monte Carlo alapú megközelítés: véletlen pontrészhalmaz között menő legrövidebb utak által érintett élek számítanak csak



Gráf klaszterezők – Modularitás

- Ötlet:
  - Véletlen gráfoknak nincs struktúrája → maximalizáljuk az ettől való eltérést
  - Modularitás: véletlen gráftól való eltérés mértéke

$$Q = \frac{1}{2m} \sum_{ij} (A_{ij} - P_{ij})\, \delta(C_i, C_j)$$

  - A – szomszédsági mátrix
  - P – szomszédsági valószínűség a véletlen gráfban
- Algoritmus:
  - Kezdetben minden csúcs külön klaszter
  - Adjuk hozzá azt az élt amelyik a legnagyobb mértékben növeli a modularitást
  - Több módosítás a fenti mérték maximalizálására: szimulált hűtés, genetikus megközelítések