Gépi tanulás

Tanulás fogalma

- Egy algoritmus akkor tanul, ha egy feladat megoldása során olyan változások következnek be a működésében, hogy később ugyanazt a feladatot vagy ahhoz hasonló más feladatokat jobb eredménnyel, illetve jobb hatékonysággal képes megoldani, mint korábban.
- A tanulással meg lehet adni a feladat
  - modelljét (logikai formulák, valószínűségi hálók)
  - megoldó algoritmusát (genetikus programozás, mély hálók)
  - heurisztikáját (B’ algoritmus)

Tanulási modellek

- Ha a megoldandó problémát egy $\varphi : X \to Y$ leképezés modellezi, akkor ehhez azt az $f : X \to Y$ leképezést kiszámító algoritmust keressük (tanuljuk meg), amelyre $f \approx \varphi$
  - sokszor egy rögzített $f : P \times X \to Y$ leképezést használunk, és annak azon $\Theta \in P$ paraméterét keressük, amelyre $f(\Theta, x) \approx \varphi(x)$
- *Induktív tanulási modell*
  - f leképezést (illetve annak paraméterét) $x_n \in X$ $(n=1..N)$ bemenetek (minták) alapján tanuljuk
- *Adaptív (inkrementális) tanulás*
  - Egy már megtanult f leképezést egy új minta anélkül módosít, hogy a korábbi mintákat újra meg kell vizsgálnunk.

Induktív modellek tanulási módjai

- *Felügyelt tanulás*: ismeri a tanuláshoz használt minták elvárt kimenetét is, azaz az $(x_n, \varphi(x_n))$ $(n=1..N)$ input-output párok alapján tanul.
- *Felügyelet nélküli tanulás*: nem ismeri a tanuláshoz használt minták elvárt kimenetét, csak x_n $(n=1..N)$ lehetséges inputokat; a minták illetve az azokra kiszámolt kimenetek közötti összefüggéseket próbálja felismerni, azokat osztályozni.
- *Megerősítéses tanulás*: nem ismeri ugyan a tanuláshoz használt minták elvárt kimenetét, de képes az x_n $(n=1..N)$ inputokra kiszámolt eredményt minősíteni, hogy az mennyire megfelelő.

1. Felügyelt tanulás

- A problémát modellező $\varphi : X \to Y$ leképezés közelítéséhez választunk egy $f : P \times X \to Y$ paraméteres leképezést, majd ennek azon $\Theta \in P$ paraméterét keressük (*paraméteres tanulás*), amelyre az (x_n, y_n) $(n=1..N)$ tanító minták mellett (ahol $y_n = \varphi(x_n)$) az alábbi $L(\Theta)$ hiba már elég kicsi (ettől reméljük, hogy $f(\Theta, x) \approx \varphi(x)$)

$$L(\Theta) = \frac{1}{N} \sum_{n=1}^{N} \ell\left(\underbrace{f(\Theta, x_n)}_{t_n}, y_n \right)$$

elvárt kimenet

számított kimenet

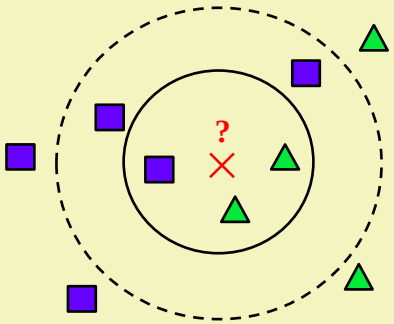
- $\ell : Y \times Y \to \mathbb{R}$ hibafüggvény (ha Y m dimenziós)
  - $\ell(t_n, y_n)$ lehet például $\|t_n - y_n\|_1$, $\|t_n - y_n\|_2^2$, vagy $-\Sigma_{j=1..m}\, y_{nj} \cdot \log t_{nj}$.

Megjegyzés

- Fontos, hogy az $f(\Theta, x)$ kiszámítása gyors legyen; nem baj, ha a megfelelő Θ megtalálása lassú, hiszen ezt a tanító minták segítségével előre számoljuk ki.
- A Θ megtanulása akkor működik jól, ha
  - N elég nagy (Ugyanakkor számolni kell azzal, hogy a mintákat drága összegyűjteni, a $\varphi(x_n)$-eket költséges kiszámolni.)
  - f és ℓ megfelelőek (ehhez tapasztalat, sok próbálkozás kell)
  - Θ közel esik a paraméter globális optimumához
- A Θ megtalálása egy nem-konvex optimalizálási feladat: a Θ globális optimumának megtalálása egy NP-teljes probléma. Szerencsére ez nem is cél, mert ezzel túl mohó módszert kapnánk (túltanulás), amely a tanító mintákra tökéletes, de egyébként nem.

1.1. *K* legközelebbi szomszéd

- Egy $x \in X$ bemenethez annak a K darab mintának az outputja alapján számol kimenetet (pl. átlagolással, vagy többségi szavazással), amely minták inputja a legközelebb esik az x-hez.



A módszer és értékelése

$sort(x, P, K)$: a $P = \{ x_n \mid n = 1..N \}$ tanító minták bemeneteiből képzett, az x-től vett távolság alapján növekvő sorozat első K eleme.

ha egy állítás igaz, akkor 1-et ad, különben 0-t

$$f(\Theta, x) = \sum_{n=1..N} \frac{\mathbb{I}(x_n \in sort(x, P, K))}{K} \cdot y_n$$

$$f(\Theta, x) = \arg\max_{n=1..N} \sum_{x_i \in sort(x, P, K),\ \varphi(x_i) = y_n} 1$$

- a Θ paraméter a minták és a $K \in \mathbb{N}$ szám együttese
- a legközelebbi szomszédokat az $\|x_n - x\|_2^2$ távolságok sorba rendezésével választjuk ki

előny: egyszerű leprogramozni, a „tanulás” gyors

hátrány: ha N nagy, a tárolás költséges;
az f kiszámítása, azaz a minták sorba rendezése erőforrásigényes

1.2. Döntési fa

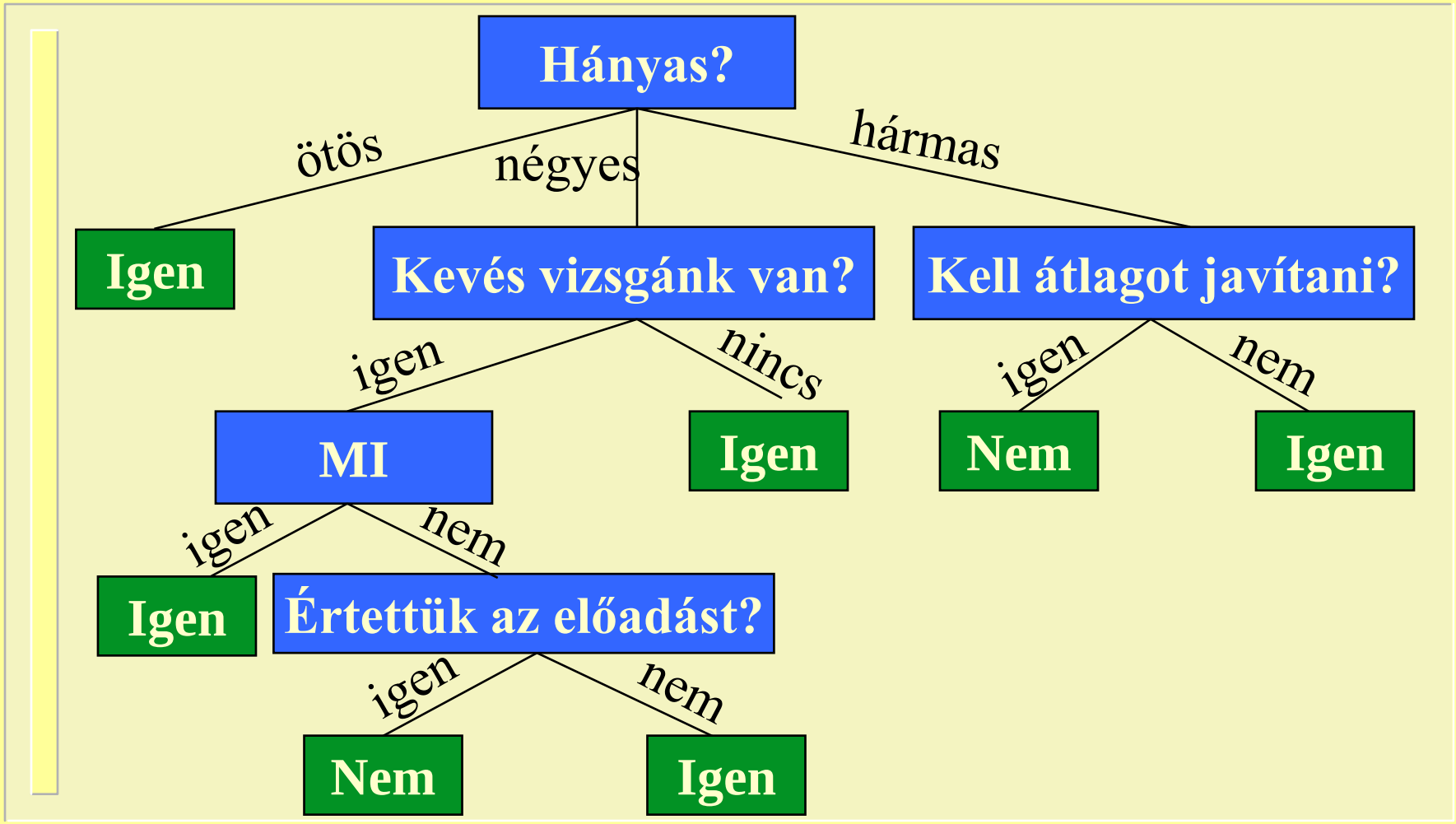
- Tegyük fel, hogy az $x \in X$ bemeneteknek ugyanazon tulajdonságait (adott attribútumainak értékeit) ismerjük, azaz egy bemenetet attribútum-érték párok halmazával jellemezhetünk.
- Képzeljük el azt az irányított fát, amelynek
  - belső csúcsai egy-egy attribútumot szimbolizálnak, és az abból kivezető éleket ezen attribútum lehetséges értékei címkézik
  - ágai attribútum-érték párok halmazát jelölik ki
  - levelei egy-egy lehetséges kimeneti értéket mutatnak
- Egy x bemenet az attribútum-érték párjai alapján egyértelműen leképezhető a döntési fa egyik levelére, amelyik a bemenethez tartozó kimenetet adja meg.

Példa: Elfogadjuk-e a megajánlott vizsgajegyet?

- Minták:
  - Ha az ötös, akkor feltétlenül.
  - Ha négyes és kevés vizsgánk van és értettük az előadást, akkor nem; feltéve, hogy a tárgy nem a Mesterséges intelligencia.
  - Ha hármas és az átlagot kell javítanunk, akkor nem.

Attribútumok és lehetséges értékeik:
- hányast ajánlottak meg (3, 4, 5)
- kevés vizsgánk van-e (igen, nem)
- kell-e átlagot javítani? (igen, nem)
- az MI tárgyról van-e szó? (igen, nem)
- értettük-e az előadást? (igen, nem)

Példa: Elfogadjuk-e a megajánlott vizsgajegyet?



Döntési fa építése

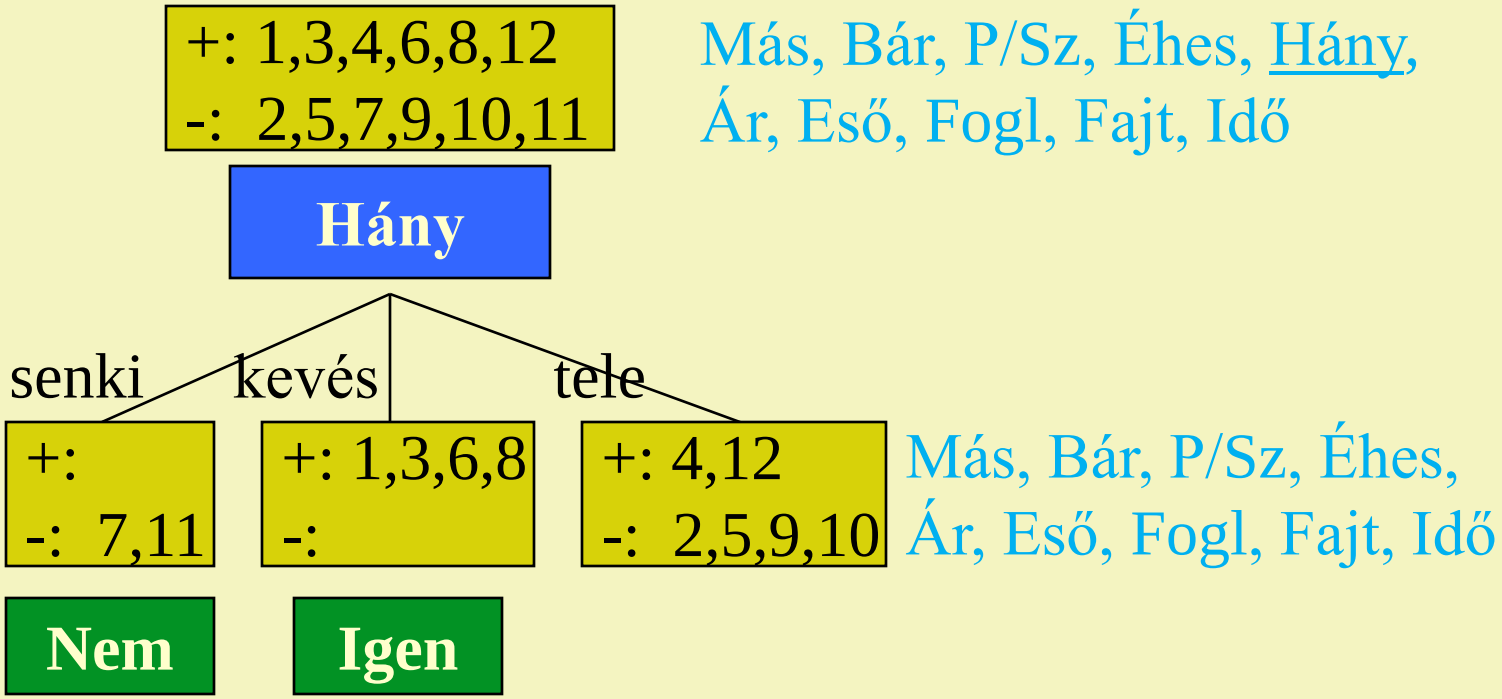
- A döntési fát tanító minták segítségével építjük fel. Ennek során
  - minden csúcshoz hozzárendeljük azon tanító mintákat, amelyek a csúcshoz vezető ág attribútum-érték párjaival rendelkeznek.
  - egy csúcs úgy válik belső csúccsá, hogy egy – a hozzávezető ágon még nem szereplő – attribútummal címkézzük fel.
  - egy csúcsot véglegesen levélcsúcsnak nyilváníthatunk, ha
    - nincsenek tanító mintái, vagy mind hasonló kimenetű, vagy a csúcshoz vezető ágon már minden attribútum szerepel.
  - a levélcsúcs értéke a hozzá tartozó tanító minták kimeneteinek átlaga vagy a leggyakoribb kimenete lesz.

    (Ha nem tartoznak minták a levélcsúcshoz, vagy nem egyértelmű, melyik a leggyakoribb kimenet, akkor a szülőcsúcsának mintái alapján számolunk.)

Étterem probléma (Russel-Norvig)

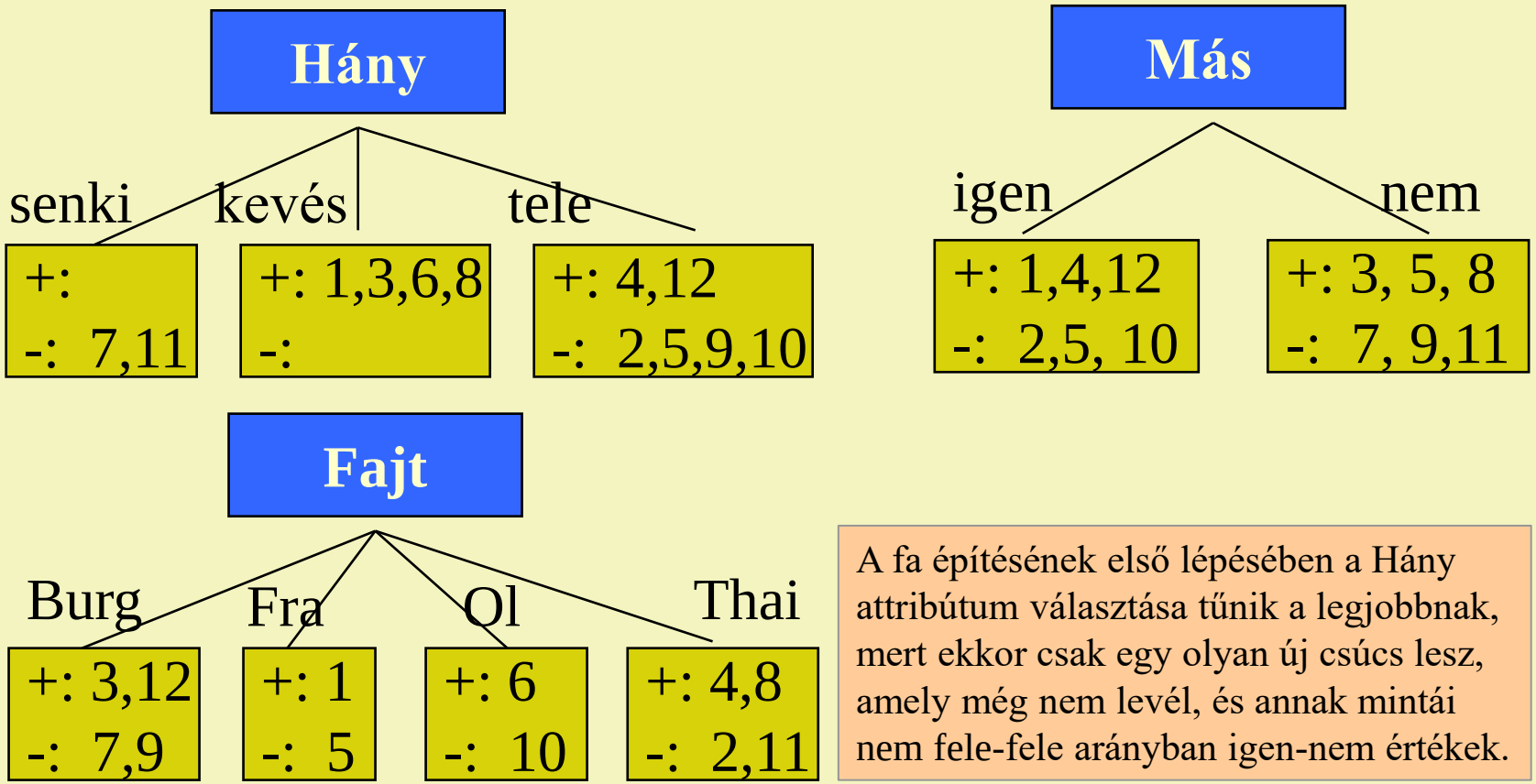
Pl.	Más	Bár	P/Sz	Éhes	Hány	Ár	Eső	Fogl	Fajt	Idő	Marad
1	I	N	N	I	kevés	drá	N	I	Fra	10	I
2	I	N	N	I	tele	olcs	N	N	Tha	60	N
3	N	I	N	N	kevés	olcs	N	N	Bur	10	I
4	I	N	I	I	tele	olcs	N	N	Tha	30	I
5	I	N	I	N	tele	drá	N	I	Fra	sok	N
6	N	I	N	I	kevés	köz	I	I	Ol	10	I
7	N	I	N	N	senki	olcs	I	N	Bur	10	N
8	N	N	N	I	kevés	köz	I	I	Tha	10	I
9	N	I	I	N	tele	olcs	I	N	Bur	sok	N
10	I	I	I	I	tele	drá	N	I	Ol	30	N
11	N	N	N	N	senki	olcs	N	N	Tha	10	N
12	I	I	I	I	tele	olcs	N	N	Bur	60	I

Döntési fa építésének első lépése



Ugyanazon problémához több döntési fa is megadható.
A lehető legkisebb döntési fa megtalálása egy NP-teljes probléma.

Alternatív lépések



A fa építésének első lépésében a Hány attribútum választása tűnik a legjobbnak, mert ekkor csak egy olyan új csúcs lesz, amely még nem levél, és annak mintái nem fele-fele arányban igen-nem értékek.

Heurisztika

- Egy döntési fa annál kisebb (annál hamarabb lehet benne egy tetszőleges bemenethez illeszkedő levélcsúcsot találni), minél rövidebbek az ágai.
- Ennek érdekében a fa építése során minél hamarabb levélcsúcsokat próbálunk meg képezni. Ehhez az kell, hogy egy csúcs tanító mintáinak kimenetei között kicsi legyen az eltérés (a 2-es norma), vagy minél kevésbé legyenek a kimentek változatosak (entrópia).
- Egy csúcshoz tehát úgy érdemes attribútumot választani, hogy megkeressük, melyik attribútum mellett kapunk a gyerekcsúcsoknál összességében legkisebb eltérésű kimenetekkel rendelkező tanító minta halmazokat.

Információ tartalom (Entrópia)

- *P*-beli minták információtartalma (entrópiája):

$$E(P) = E(p_1, \ldots, p_n) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

  - ahol a *P*-ben *n* féle eltérő értékkel rendelkező minta van, amelyek *P*-beli gyakoriságainak aránya $p_1 : \ldots : p_n$, és $p_1 + \ldots + p_n = 1$

- Az éttermes problémában a mintáknak mindössze kétféle (pozitív vagy negatív) kimenete lehet. Ekkor

$$E(P) = E(p^+, p^-) = -p^+ \log_2 p^+ - p^- \log_2 p^-$$

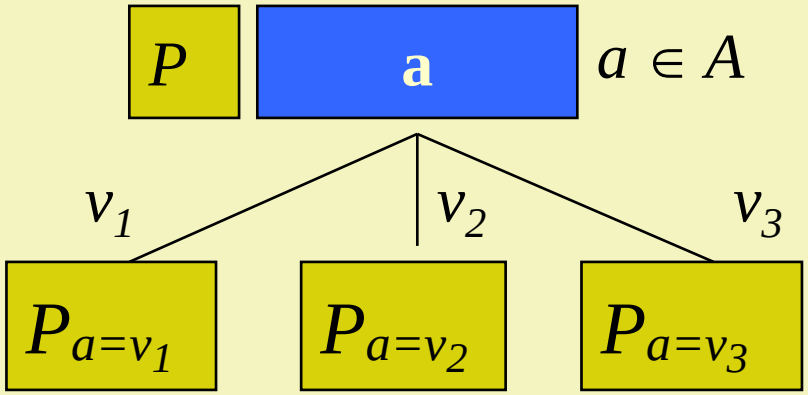
  - ahol p^+ a *P*-beli pozitív, p^- a negatív minták aránya ($p^+ + p^- = 1$)

    - Példa:

      Ha *P*-ben 2 pozitív és 3 negatív minta van: $E(P) = E(2/5, 3/5) = 0.97$

      Ha *P*-ben 0 pozitív és 3 negatív minta van: $E(P) = E(0/3, 3/3) = 0$

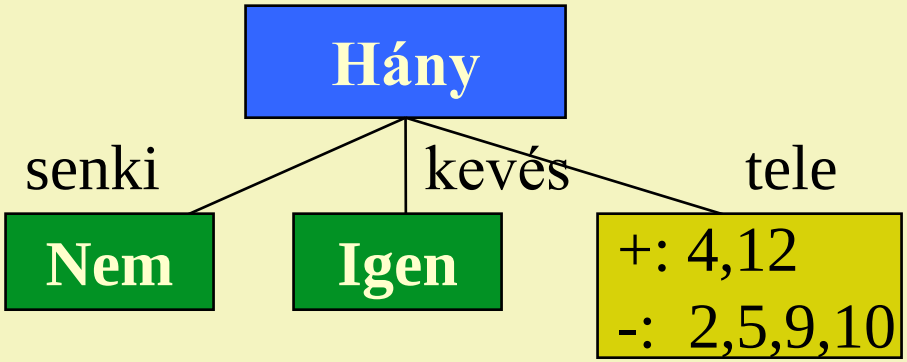
Információs előny számítása



$$C(P,a) = E(P) - \sum_{v \in Érték(a)} \frac{|P_{a=v}|}{|P|} E(P_{a=v})$$

- ahol P a szülő csúcs mintái, a a választott attribútum,
- az $Érték(a)$ az a attribútum által felvett értékek, és
- a $P_{a=v} = \{ p \in P \mid p.a=v \}$

Egy csúcs attribútumának kiválasztása 1.



$E(\{2,4,5,9,10,12\}) =$
$= E(2/6,4/6) = 0.92$

Más, Bár, P/Sz, Éhes,
Ár, Eső, Fogl, Fajt, Idő

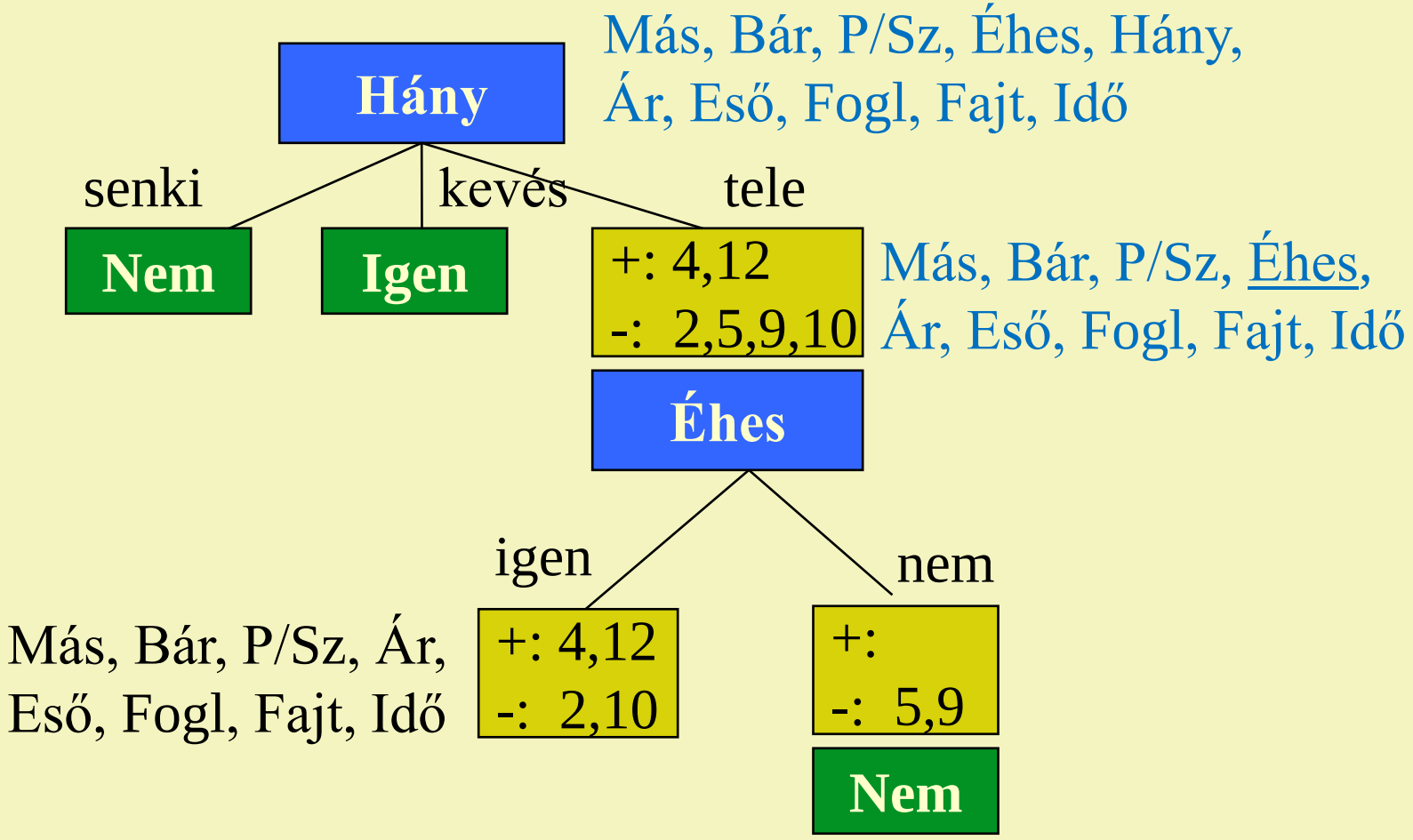
- Ha a fa építés folytatásakor a *Más* attribútumot választjuk, akkor a minták 1:5 arányban ketté válnak: {9} (*Más= hamis*), és {2, 4, 5, 10, 12} (*Más=igaz*),
  - $E(\{9\}) = E(0/1, 1/1) = 0$
  - $E(\{2,4,5,10,12\}) = E(2/5, 3/5) = 0.97$
- Az információs előny: $C(\{2,4,5,9,10,12\}, Más) = E(\{2,4,5,9,10,12\}) - (1/6\ E(\{9\}) + 5/6\ E(\{2,4,5,10,12\})) =$
  $E(2/6,4/6) - (1/6\ E(0/1,1/1) + 5/6\ E(2/5,3/5)) = 0.92 - 0.81 = 0.11$

Egy csúcs attribútumának kiválasztása 2.

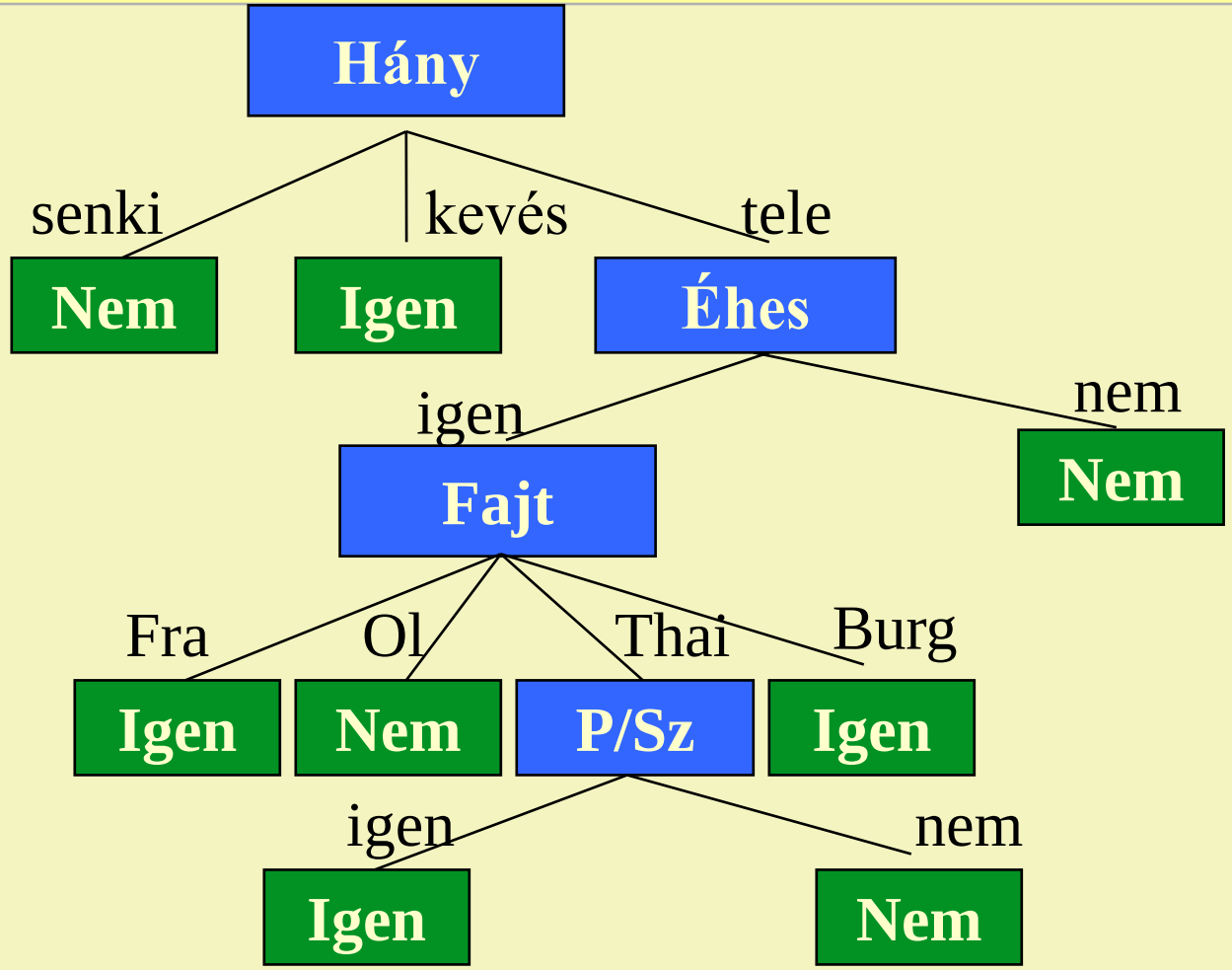
$C(\{2,4,5,9,10,12\},a) = 0.92$ -

Más:	$1/6\ E(0/1,1/1)+5/6\ E(2/5,3/5)=$	*0.81*
Bár:	$3/6\ E(1/3,2/3)+3/6\ E(1/3,2/3))=$	*0.92*
P/Sz:	$1/6\ E(0/1,1/1)+5/6\ E(2/5,3/5)=$	*0.81*
Éhes:	$4/6\ E(2/4,2/4)+2/6\ E(0/2,2/2)=$	*0.67*
Ár:	$4/6\ E(2/4,2/4)+0/6\ E(0,0)+\ 2/6\ E(0/2,2/2)=$	*0.67*
Eső:	$5/6\ E(2/5,3/5)+1/6\ E(0/1,1/1)=$	*0.81*
Fog:	$4/6\ E(2/4,2/4)+2/6\ E(0/2,2/2)=$	*0.67*
Fajt:	$2/6\ E(1/2,1/2)+1/6\ E(0/1,1/1)+1/6\ E(0/1,1/1)+2/6\ E(1/2,1/2)=$	*0.67*
Idő:	$0/6\ E(0,0\ )+2/6\ E(1/2,1/2)+2/6\ E(1/2,1/2\ )+2/6\ E(0/2,2/2)=$	*0.67*

További lépések



Étterem probléma döntési fája



Készítsünk algoritmust

- Egy fokozatosan épülő döntési fában minden csúcsnál eltároljuk
  - a csúcshoz tartozó (csúcshoz vezető ág attribútum-érték párjaival rendelkező) tanító mintákat
  - a csúcsnál még választható (a csúcshoz vezető út csúcsainak címkéiben nem szereplő) attribútumokat
- Egy csúcs lehet
  - attribútummal címkézett belső csúcs, amelyekből kivezető éleket az attribútum lehetséges értékei címkézik
  - kiértékelt vagy értékkel nem rendelkező levélcsúcsok
- Minden lépésben egy értékkel még nem rendelkező levélcsúcsról kell eldönteni, hogy kaphat-e értéket vagy legyen-e belső csúcs.

Algoritmus

- Kezdetben a fa egyetlen címkézettlen csúcsból áll (ez lesz majd a gyökér), amelyhez az összes mintát és attribútumot rendeljük.
- Veszünk a fából egy értékkel még nem rendelkező levélcsúcsot, amíg van ilyen:
  1. Ha a csúcsnak nincsenek mintái ($P=\varnothing$), akkor a szülőcsúcsának mintái alapján kiszámoljuk a csúcs értékét.
  2. Ha a csúcshoz tartozó minták kimenete nem nagyon tér el egymástól, akkor ezekből kiszámoljuk a csúcs értékét.
  3. Ha a csúcsnak nincsenek választható attribútumai ($A=\varnothing$), akkor a csúcs mintái alapján kiszámoljuk a csúcs értékét.
  4. Egyébként a választható attribútumok közül a legkedvezőbbnek látszóval megcímkézzük az adott csúcsot, és generáljuk a gyerekeit az azokhoz tartozó mintákkal és választható attribútumokkal együtt.

Megjegyzés

- Zaj: Azonos attribútum-értékű minták kimenete különbözik.
  - Ilyenkor a minták válaszainak átlagolása félrevezethet
- Túlzott illeszkedés: A bemenetek olyan attribútumait is figyelembe veszünk, amelyek a kimenetre nincsenek hatással. (Például egy kocka dobás eredményére a kocka színe és a dobás dátuma alapján értelmetlen szabályszerűségeket találunk.)
  - A lényegtelen attribútumokat ($C(P,a) \sim 0$) állítsuk félre.
- Általánosítások:
  - Hiányzó adatok (attribútum értékek) problémája
  - Folytonos értékű attribútumok

Tanulás döntési fával

- Egy döntési fában az $x \in X$ bemenetre kiszámolt levélcsúcs értéke:

$P(x)$ az x-re kiszámolt levélcsúcs tanító mintái bemeneteinek halmaza

$$f(\Theta, x) = \sum_{n=1..N} \frac{\mathbb{I}(x_n \in P(x))}{|P(x)|} \cdot y_n$$

$$f(\Theta, x) = \arg\max_{n=1..N} \sum_{\substack{x_i \in P(x) \\ \varphi(x_i) = y_n}} 1$$

  - Θ a döntési fa, amely optimalizálása annak mohó felépítése.

előny: jól értelmezhető (a mintákra tökéletes eredményt ad, és a mintákhoz hasonló inputokra többnyire jó eredményt ad); a tanító minták helyett csak a döntési fát kell tárolni; x-re adott eredmény gyorsan számolható

hátrány: az optimális döntési fa építése NP-teljes, a bemutatott mohó módszerrel csak lokálisan optimális döntési fához jutunk

1.3. Véletlen erdő

- K darab döntési fát építünk a tanító minták alapján úgy, hogy egy-egy fa építéséhez a tanító mintáknak is ($P_k \subseteq P$), és az attribútumoknak is ($A_k \subseteq A$) csak egy-egy véletlen kiválasztott részhalmazát használjuk fel. Így kapjuk a véletlen erdőt.
- Egy véletlen erdő minden fájában külön-külön megállapíthatjuk, hogy egy $x \in X$ bemenet a döntési fa melyik levelére képződik le. Ezen levelekhez tartozó tanító mintahalmazok kimenetei alapján becsüljük az x kimenetét. Ez lehet az egyes fák
  - adott levelei értékéből képzett átlag, vagy
  - adott leveleinek értékei közül a leggyakoribb

Tanulás véletlen erdővel

- Egy *x* bemenethez tartozó kimenetet a minták kimeneteinek súlyozott átlaga, ahol a súlyok attól függenek, hogy egy minta a véletlen erdő döntési fáinak *x*-re kiszámolt levélcsúcsaihoz tartozó mintahalmazok közül hányba esik bele, és az a halmaz hány elemű:

$$f(\Theta, x) = \frac{\sum_{k=1}^{K} \text{value}(\, x, \text{decision\_tree}(A_k, P_k)\,)}{K}$$

ami a P_k és az A_k alapján felépített döntési fából számolható ki *x*-re

$$f(\Theta, x) = \arg\max_{k=1}^{K} \sum_{i=1}^{K} \mathbb{I}(\text{value}(x, \text{decision\_tree}(A_k, P_k)) = \text{value}(x, \text{decision\_tree}(A_i, P_i)))$$

  - Θ maga a véletlen erdő, optimalizálása az erdő felépítése

előny: a tanító minták helyett csak az erdőt kell tárolni;
a véletlen generálás miatt kevésbé mohó, elkerüli a túltanulást;
az *x*-re adott eredmény számolása párhuzamosítható

hátrány: az eredmény kevésbé értelmezhető; az erdő-építés NP-teljes

2. Felügyelet nélküli tanulás

- Ismertebb módszerek:
  - Klaszterezés
  - Dimenzió csökkentés
  - Autóenkóderek

1.1. *k*-közép módszer

- Legyenek a klaszterezendő (x_p) elemek $\mathbb{R}^n$-beli pontok. Soroljuk be ezeket k darab klaszter valamelyikébe.
  - Minden klasztert (S_i) a középpontjával (m_i) reprezentálunk, ami a klaszterhez tartozó pontok átlaga (centroid). $m_i = \frac{1}{|S_i|} \sum_{x_j \in S_i} x_j$
- Keressük azt a k klasztert (középpontok és hozzá tartozó elemek), amelyre minimális az elemeknek a klaszterük középpontjától számított távolságnégyzeteinek összege.
  - Ez ekvivalens a klaszteren belüli pontoknak a páronkénti távolságnégyzetük minimalizálásával.

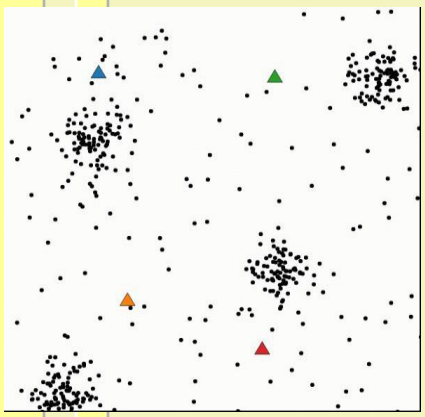
$$\arg\min_{\boldsymbol{S}} \sum_{i=1}^{k} \sum_{x \in S_i} \|x - m_i\|_2^2 = \arg\min_{\boldsymbol{S}} \sum_{i=1}^{k} \frac{1}{2 \cdot |S_i|} \sum_{x,y \in S_i} \|x - y\|_2^2$$

k-közép módszer algoritmusa

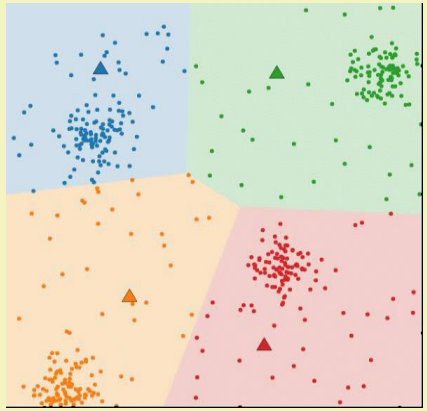
- Kezdetben adott a k és $\mathbb{R}^n$-beli x_p elemek
- Inicializáljuk a k darab centroidot: $m_1^{(1)}, \ldots, m_k^{(1)}$
  - Vagy véletlenszerűen választunk k középpontot,
    vagy minden adatpontot véletlenszerűen egy-egy klaszterbe sorolunk
    és kiszámoljuk azok középpontjait.
- Az alábbi lépések a középpontok t-edik változatát $t+1$-dikre cseréli:
  1. Az elemeket a legközelebbi klaszter középponthoz rendeljük:
     $$S_i^{(t+1)} = \{ x_p \mid \forall j \in [1 \,..\, k] : \|x_p - m_i^{(t)}\|_2^2 \leq \|x_p - m_j^{(t)}\|_2^2 \}$$
  2. Kiszámítjuk az új elemcsoportok középpontjait:
     $$m_i^{(t+1)} = \frac{1}{|S_i^{(t+1)}|} \sum_{x_j \in S_i^{(t+1)}} x_j$$
- Addig számolunk, amíg két lépés eredménye (középpontjai) közötti eltérés adott határ alá nem kerül.

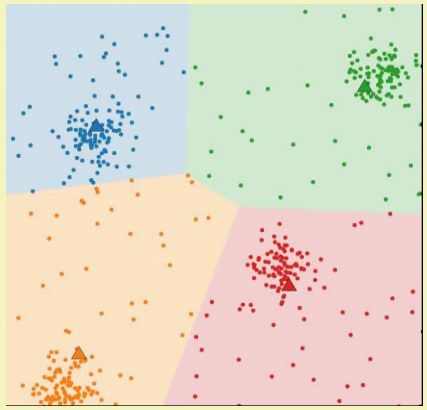
Példa a k-közép módszer működésére

inicializálás

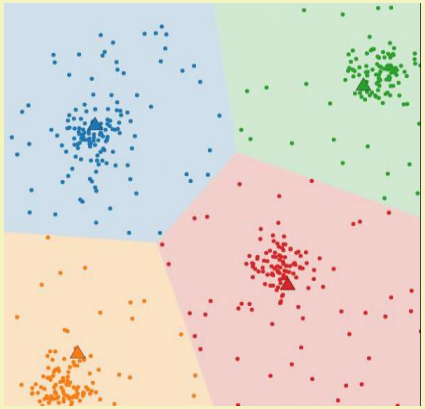


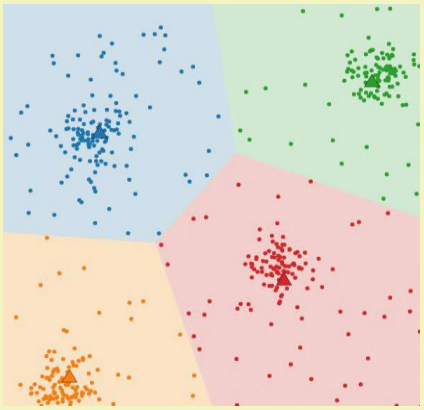
1. menet



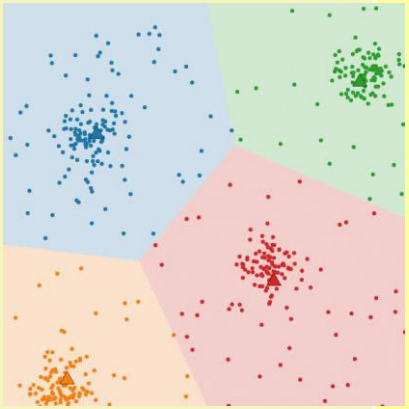


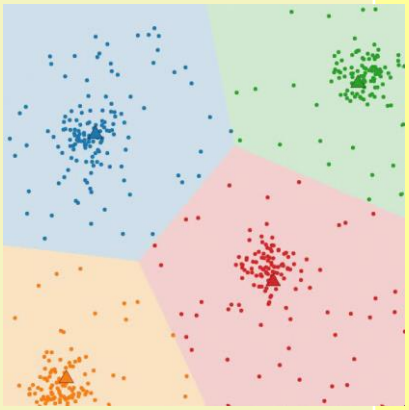
2. menet





3. menet





Megjegyzés

- A *k*-közép egy kemény klaszterező eljárás (*hard clustering)*.
- A klaszterek megtanulása után bármelyik $\mathbb{R}^n$-beli pont besorolható a meglévő klaszterek közül a legközelebbibe. Természetesen itt sem mindegy az, hogy a tanító példák mennyire jól reprezentálják az X halmaz elemeit.
- Tulajdonképpen itt egy olyan $\varphi : X \to Y$ leképezést tanulunk meg, amelyik az X-beli tetszőleges elemeket (nemcsak a tanító halmaz elemeit) a megtanult k klaszter egyikébe sorolja, azaz osztályozza az az X-beli elemeket. ($Y = [1..k]$)

Megjegyzés

- Az optimális klaszterek megtalálása NP-nehéz probléma: ezért alkalmazunk approximációt, de a módszer csak lokális optimumot biztosít.
- Problémát jelent, ha a megadott *k*
  - túl nagy: ekkor kialakulnak nagyon kevés elemet tartalmazó „üres” klaszterek
  - túl kicsi: ekkor egy klaszteren belül lehetnek szignifikánsan elkülönülő csomósodások
- A módszert többször futtathatjuk különböző véletlen inicializációkkal.