Mesterséges intelligencia II.

Felügyelet nélküli tanulás

A gépi tanulás célja: A gépi tanulás célja olyan programok készítése, amelyek a működés során szerzett tapasztalatok alapján képesek javítani saját hatékonyságukon.

Tanuló-algoritmusok: Olyan algoritmusok, melyek képes összefüggések, szabályosságok megtanulására (hipotetizálására) egy példa-halmaz alapján. Ezt a példahalmazt jelöljük X-szel.

Induktív tanulás: A tanítás során nem látott esetekre is szeretnénk általánosítani a megtanult összefüggéseket. Feltevés: A példák hűen reprezentálják a tanulandó összefüggést.

Felügyelet mélküli tanulás: Felügyelet nélküli tanulásnak nevezzük, ha a példákhoz nincs „külső segítség", azaz nem adottak az osztálycímkék. Leggyakoribb feladat a **klaszterezés:** adatok automatikusan kialakított osztályokba sorolása valamilyen hasonlóság, vagy különbözőség alapján.

Klaszterező algoritmusok használata közben általában előre definiált a klaszterek száma. Ha nem definiált a klaszterek száma, akkor a hierarchikus klaszterezés lehet az egyik lehetséges megoldása a feladatnak.

Hierarchikus klaszterezők: A cél a példák halmaza felett egy hierarchia definiálása, azaz egy fa építése melynek levelei reprezentálják az input példákat ($\in X$) és a fa bármely belő pontja két diszjunkt klaszterre (halmazra) bontja a levelek halmazát. Két módszert különböztetünk meg a felülről lefelé és alulról felfelé építő algoritmusok. Az alulról felfelé építők terjedtek el, ugyanis ezek a levelek összevonásával építik a fát. Elterjedésük oka: egyszerűbb, logikusabb, gyorsabb és jobban teljesítenek.

A példák közötti távolság (eltérés) egy nXn-es távolságmátrixszal (D) reprezentálható. Ebben a mátrixban az ij-edik elem (d_{ij}) az i-edik példa távolságát jelenti a j-edik példától.

A fa optimális ha a négyzetes hiba minimális (nulla).

$$Err = \sum_{i,j \in X} (dt_{ij} - d_{ij})^2 \rightarrow min$$

dt_{ij} az i és j csúcs **fatávolsága**, d_{ij} az i és j példák távolsága. A fa oprimális ha a négyzetes hiba minimális. Egynégyzetes függvény minimuma ott van ahol az első deriváltjának az értéke O. Azaz a fa optimális, ha $dt_{ij} = d_{ij}$.

Általános alulról építő algoritmus:

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 $|C| \neq 1$.

A faérítő algoritmusok általában a *távolság újradefiniálása* lépésben térnek el egymástól.

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.