A legközelebbi szomszéd osztályozási módszer algoritmikus problémáiról

DR. FARAGÓ ANDRÁS—LINDÉR TAMÁS—
PIKLER TAMÁS—LUGOSI GÁBOR
BME Híradástechnikai Elektronika Intézet

ÖSSZEFOGLALÁS

A cikkben két új módszert mutatunk be a döntési és osztályozási feladatokban elterjedten használt „legközelebbi szomszéd" módszer algoritmikus felgyorsítására. Az egyik úgy csökkenti a szükséges távolságszámítások számát, hogy közben az eredményt változatlanul hagyja, tehát a gyorsítás nem jár együtt a hibavalószínűség növekedésével. A másik módszer egy speciális clusterezési eljárás, amely két- ill. többszintű döntést tesz lehetővé. Az eredményeket matematikai ós szimulációs úton egyaránt igazoljuk.

Bevezetés

Az alakfelismerésben és más területeken is (pl. vektorkvantálás) az egyik legelterjedtebb osztályozási eljárás a *legközelebbi szomszéd módszer* (Nearest Neighbour, rövidítve NN). Ennek lényege roppant egyszerűen megfogalmazható: a felismerendő (azaz osztályozandó) pontot abba a kategóriába soroljuk, amelynek valamely reprezentáns pontja (más néven tanulópont, mert a rendszer „betanításához" használjuk) a felismerendő ponthoz a legközelebb esik az összes tanulópont közül, valamilyen — a feladattól függő — távolságmérték szerint. E — rendszerint sokdimenziós térben reprezentált — pontok azután bármilyen valóságos felismerendő objektum adatait hordozhatják.

A módszer alkalmazásaiban gyakran előfordul, hogy a pontok csak többszáz dimenziós vektorokkal adhatók meg, és a tanulópontok száma is százas nagyságrendű (vagy még több). Ez a helyzet pl. a gépi beszédfelismerésben is. Az ilyen alkalmazásokban szinte kínzó szükségszerűséggel merül fel az eljárás algoritmikus felgyorsításának igénye, mert enélkül a real-time megvalósításra kevés az esély.

Az alábbiakban két új módszert mutatunk be az NN osztályozás felgyorsítására. Az első úgy éri el ezt a gyorsítást, hogy ugyanakkor az NN módszerrel garantáltan azonos eredményt biztosít (tehát nem áldozza fel a pontosságot a sebességért, mint az irodalomban található legtöbb eljárás: [1], [2], [4], [6], [8]). A gyorsítás ára a megnövekedett tárigény és előfeldolgozás. Ezt az árat azonban érdemes megfizetni olyan alkalmazásokban, ahol kritikus a valós idejű működés.

A második módszercsoport tulajdonképpen clusterezési eljárásokat mutat be, amelyek már adhatnak eltérő eredményt az NN módszerhez képest, de bizonyos feltételek mellett ennek kicsi a valószínűsége.

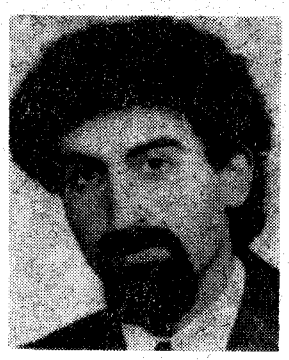
Az NN osztályozással kapcsolatos kutatásainkhoz a gyakorlati motivációt a Budapesti Műszaki Egyetem Híradástechnikai Elektronikai Intézetében (BME—HEI) fejlesztés alatt álló VERBIDENT—1 izolált szavas gépi beszédfelismerő adta. Mivel eredményeink azonban más területeken is felhasználhatók, ezért általános formában írjuk le, nem kötődve a gépi beszédfelismerés specialitásaihoz. A módszereknek itt csak a lényegét ismertetjük, a matematikai bizonyítások elolvashatók [10]-ben.

NN osztályozás folyamatos szelekcióval

Most az NN-algoritmus egy olyan meggyorsítását mutajuk be, amely a pontosságból nem áldoz fel semmit, azaz eredménye megegyezik az eredeti NN osztályozásával.

Tegyük fel, hogy a C_1, C_2, ..., C_N kategóriákat az egyszerűség kedvéért egy-egy tanulópont képviseli. A t_1, ..., t_N tanulópontok és az x osztályozandó pont az M metrikus tér elemei, amelyen a d távolságfüggvény van értelmezve. Az NN-döntés

Beérkezett: 1988. III. 2 (H)

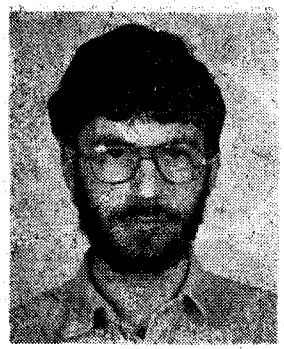


DR. FARAGÓ ANDRÁS

1976-ban szerzett villamosmérnöki oklevelet a BME Villamosmérnöki Karának Híradástechnikai szakán. 1982-ig ugyanott a Matematikai Tanszéken dolgozott tanársegédként, majd a BME Híradástechnikai Elektronika Intézetébe került, ahol jelenleg adjunktusként dolgozik az Átvitel- és Rendszertechnika Osztályon. A villamosmérnök hallgatók oktatása mellett fő szakmai érdeklődési és kutatási területe a digitális beszédfeldolgozás. Ezenkívül foglalkozik matematikai kutatással is, a gráfelmélet és kombinatorikus optimalizálás területén.

LINDER TAMÁS

A BME Villamosmérnöki Kar Híradástechnikai szakának IV-éves hallgatója. 1986 óta az MTA Akusztikai Kutatólaboratóriumban beszédfelismerő rendszer fejlesztésében vesz részt. A HTE tagja.



PIKLER TAMÁS

A BME Villamosmérnöki Kar Híradástechnikai szakának végzős hallgatója. 1985 óta az MTA SZTAKI-ban multiprocesszoros rendszer fejlesztésében vesz részt. A HTE tagja.

LUGOSI GÁBOR

A BME Villamosmérnöki Kar Híradástechnikai szakának végzős hallgatója. 1986 óta az MTA Akusztikai Kutatólaboratóriumban beszédfelismerő rendszer fejlesztésében vesz részt. A HTE tagja.

szerint x osztálya akkor C_k, ha $d(x,t_k) \leq d(x,t_i)$ $\forall(i \in \{1,\ldots,N\}$; egyenlőség esetén a kisebb indexűt választjuk). Célunk tehát az x-től legkisebb távolságra lévő tanulópont megkeresése, amely N db távolságszámítást jelent. Mivel azonban a műveletigényt a távolságszámítások száma határozza meg, az algoritmust csak úgy gyorsíthatjuk, ha nem minden tanulópontot vizsgálunk meg. Ezt úgy érjük el, hogy olyan tanulópontokat keresünk, amelyek biztosan nem lehetnek legközelebbi szomszédok, így kizárhatók a további vizsgálatokból Látható, hogy ezzel az NN módszer hibáját nem növeljük.

A vizsgálat sorrendje

Induljunk ki abból, hogy az összes tanulópont távolságát kiszámítjuk az x-től, de a vizsgálat sorrendjét válasszuk úgy, hogy a t^{NN} legközelebbi szomszéd minél előbb sorra kerüljön. Ez persze önmagában nem jelent megtakarítást, hiszen t^{NN} megvizsgálásakor még nem tudjuk, hogy nem lesz-e egy x-hez még közelebbi tanulópont, és így nem állhatunk le a további vizsgálattal. A későbbiekben azonban látni fogjuk, hogy ez sokszor megtehető.

Tegyük fel, hogy a $t_1,\ldots,t_k$ tanulópontokat már megvizsgáltuk, és tekintsük az

$$f_k(t) = \sum_{i=1}^{k} |d(t_i, t) - d(t_i, x)| =$$

$$= f_{k-1}(t) + |d(t_k, t) - d(t_k, x)|$$

függvényt. Ezt minden hátralévő $t_{k+1},\ldots, t_N$ tanulópontra kiszámítjuk, és azt a t pontot választjuk ki a következő megvizsgálandó tanulópontnak, amelyre $f_k(t)$ minimális. Ez szemléletesen azt jelenti, hogy ez a tanulópont a $t_1,\ldots, t_k$ pontoktól körülbelül olyan távolságra lesz, mint az osztályozandó pont, tehát várhatóan annak a közelébe esik. E választást jól jellemzi a következő tétel (a pontos megfogalmazással ós a bizonyítással kapcsolatban [11]-re utalunk): egy n dimenziós euklideszi térben az f(t) függvények segítségével csupán n+2 távolságszámítással olyan tanulóponthoz jutunk, amelynek osztályára döntve tetszőlegesen megközelíthetjük az NN-döntós pontosságát, ha N elég nagy. Tehát a ténylegesen elvégzendő távolságszámítások száma (aszimptotikusan) csak a dimenziószámtól függ és rögzített dimenzió esetén a tanulópontok számának növelésével egy bizonyos határ felett lényegében már nem növekszik. (Ennek persze ára van: előre ki kell számítani a tanulópontok egymástól mórt távolságát és ezt tárolnunk kell. Ez az előfeldolgozás azonban nem számít bele a real-time műveletigénybe.)

Kizárás vizsgálat közben

Újra tegyük fel, hogy a vizsgálatban t_k-nál tartunk Ekkor a még hátralévő $t_{k+1},\ldots, t_N$ tanulópontok közül nem lehetnek legközelebbi szomszédok azok a t tanulópontok, amelyekre $d(t_k,x)+r_{min} \leq d(t_k,t)$ vagy $d(t_k,x)-r_{min} \geq d(t_k,t)$, ahol r_{min} jelöli az eddig talált legközelebbi szomszéd távolságát x-től (bizonyítását lásd [10]). Ennek alapján a tanulópontok megvizsgálása után kizárásokat tehetünk, amelyek várhatóan annál hatékonyabbak, minél kisebb r_{min}. A legközelebbi szomszéd x-től való távolságának ismeretében tehát már csak a $2d(t^{NN},x) \leq d(t^{NN},t)$ tulajdonságú t tanulópontokat kell megvizsgálni.

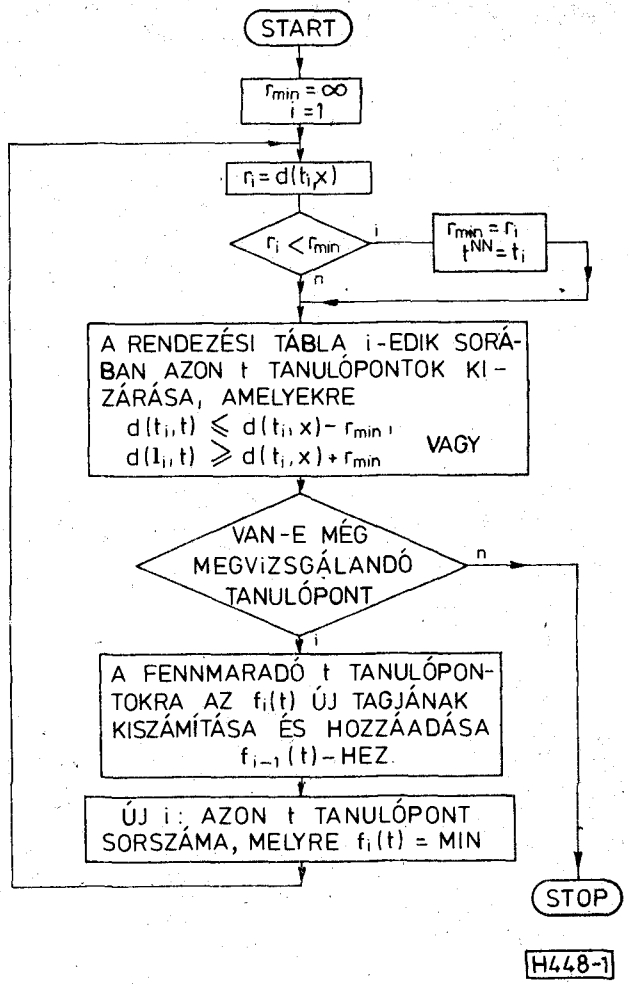
Az algoritmus programozása

Az eddigiekből látszik, hogy szükségünk van a tanulópontok egymás közötti távolságaira. Ezeket még az osztályozandó pont ismerete nékül kiszámíthatjuk és egy jól kezelhető struktúrában, az ún. rendezési táblában tároljuk. A rendezési tábla egy N sorból álló mátrix, amelynek i-edik sorában a tanulópontok sorszámait és t_i-től való távolságait soroljuk fel e távolság szerint növekvő sorrendben. A tárigény sok tanulópontnál tekintélyes, azonban nyilvánvaló, hogy a gyorsításért valamilyen árat fizetnünk kell.

A teljes algoritmus vázlatos folyamatábrája az 1. ábrán látható. Ebből kitűnik, hogy egy-egy tanulópont vizsgálata után maximálisan 4N összeadás nagyságrendjébe eső számításra van szükség a kizárások elvégzésére és a következő tanulópont kiválasztására. (A műveletigény azonban ennél kisebb, hiszen a kizárt ill. már megvizsgált tanulópontokra e mellékszámítások nem terjednek ki.) Ez a „bonyolítás" bőségesen megtérül olyankor, ha a távolságok kiszámítása nagy műveletigényű a távolságmérték bonyolultsága miatt.

Szimulációs eredmények

Az algoritmusunk hatékonyságának illusztrálására számítógépes szimulációkat készítettünk. Ennek során N db tanulópontot generáltunk egyenletes eloszlással egy n dimenziós vektortérben, majd sok osztályozandó pontra elvégeztük a döntési eljárást, és az átlagosan szükséges távolságszámítá-



1. ábra. NN osztályozás folyamatos szelekcióval, folyamatábra

sok számát vizsgáltuk. Az osztályozandó pontot egy véletlenszerűen választott tanulópontra ültetett normális eloszlású zajvektorral szimuláltuk. E zajvektor szórását a többi paraméterrel együtt a szimuláció során változtattuk.

A szimulációs eredményeket a 2. ábrán foglaltuk össze. Ezen a dimenziószám függvényében a szükséges távolságszámítások számának arányát láthatjuk százalékban. A két szélső görbe az általunk alkalmazott két szélső szórásértéknél kapott eredményeket jelzi, a közbülső szórásoknál a görbék közötti területre adódtak az értékek.

A kapott eredmények összefoglalásaként három fő megállapítást tehetünk. A legfontosabb az, hogy kb. 30%-nál rosszabb hatékonyságot még a legkedvezőtlenebb esetben sem kaptunk, tehát valóban lényeges gyorsításról van szó. Másrész ha az osztályozandó pont elég közel van valamelyik tanulóponthoz, akkor mindössze néhány távolságszámítás elegendő. Továbbá nagyobb dimenziószám felé haladva nem észlelünk hatékonyságromlást, és a töréspont körülbelül a tanulópontok számánál figyelhető meg.

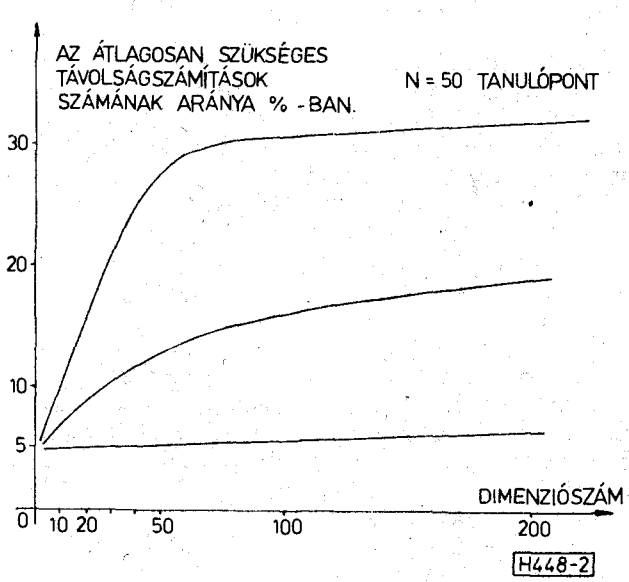
Az ismertetett elméleti és szimulációs eredmények alapján ajánljuk a módszer alkalmazását, különösen olyan esetekben, ahol a távolságfüggvény számítása nagy műveletigényű, és az osztályozandó pont elég közel esik valamelyik tanulóponthoz.

Dekompozíciós feladatok

A következőkben leírt két algoritmus lényegében két clusterezési eljárás. Clusterezésen a tanulópontok diszjunkt halmazokra való particionálását értjük. A legismertebb ilyen eljárás a dinamikus clusterezés k-középmódszere ([12]). A tanulópontok clusterekbe való particionálása úgy gyorsítja a legközelebbi szomszéd megtalálását, hogy első lépésként valamilyen módon eldöntjük azt, hogy a legközelebbi szomszéd melyik cluster eleme, és ezután csak ezen cluster elemeitől kell távolságokat számítani. Az általunk adott clusterezési algoritmus lényege az, hogy miután definiáltuk azt, hogy milyen tulajdonságú pontok kerülhetnek egy clusterbe, az algoritmus a clusterek konstruálását, számuk meghatározását maga végzi úgy, hogy a legközelebbi szomszéd megtalálásához szükséges távolságszámítások várható száma minimális legyen. E célból használjuk az önfüggő blokk fogalmát ([3]), mely elég kötött ahhoz, hogy a tanulópontok sűrűsödését várhatóan jól jellemezze.

A módszer alapját a tanulópontok T halmazának ún. önfüggő blokkokra való particionálása képezi. Az önfüggő blokk fogalmát Győrfi Zoltán vezette be 1974-ben ([3]). Definíció szerint egy $K \subseteq T$ halmazt akkor nevezünk önfüggő blokknak, ha bármely két K-beli tanulópont távolsága kisebb, mint bármely K-beli tanulópontnak akármelyik K-n kívüli tanulóponttól való távolsága. A T halmaz egy diszjunkt önfüggő blokkokra való particionálását dekompozíciónak nevezzük.

Ha adott a T halmaz egy dekompozíciója, akkor a döntési szabály legyen a következő: jelöljük ki minden önfüggő blokk egy-egy pontját — ezeket blokk-reprezentáns pontnak nevezzük —, és az



2. ábra. Szimulációs eredmények

első lépésben számítsuk ki ezeknek az osztályozandó ponttól való távolságát. Ezek után az osztályozandó pontnak csak azon önfüggő blokk elemeitől való távolságát kell kiszámítanunk, amelynek blokk-reprezentánsára ez a távolság a legkisebb volt.

Ez a döntési szabály adhat némi hibát az NN-döntóshez képest, azonban biztos, hogy ez a hiba nulla, ha tudjuk, hogy az osztályozandó pont csak valamelyik tanulópont lehet (ez előfordul pl. a keresési feladatokban), illetve kicsi, ha az osztályozandó pont nagy valószínűséggel valamelyik tanulópont közelébe esik.

Optimális dekompozíció

Egy adott T halmazhoz több dekompozíció tartozik, hiszen önfüggő blokkot alkotnak például a csupán az egyes tanulópontokat tartalmazó halmazok, de maga a T halmaz is.

Célunk az, hogy megkonstruáljuk a T halmaznak azt a dekompozícióját, amely a döntést a legjobban meggyorsítja, vagyis amelyre legkisebb a döntés meghozatalához szükséges távolságszámítások számának várható értéke.

A továbbiakban feltesszük, hogy a tanulópontok illetve az osztályozandó pont eloszlására teljesül a következő blok-egyenletességi feltétel: bármely dekompozícióban minden önfüggő blokkra annak a valószínűsége, hogy az osztályozandó pont döntésünk szerint az önfüggő blokkhoz tartozik: n/N, ahol N a T halmaz, n pedig az illető önfüggő blokk elemszáma. Ezután könnyen belétható a következő állítás: ha teljesül a blokk-egyenletességi feltétel, akkor a szükséges távolságszámítások számának várható értéke egy $\pi=\{K_1, K_2, \ldots, K_k\}$ dekompozícióra

$$m(\pi)=k-1+\frac{1}{N}\sum_{i=1}^{k} n_i^2$$

ahol n_i a K_i önfüggő blokk elemszáma ($i=1, 2, \ldots k$,) (bizonyítása megtalálható [10]-ben).

Megjegyzés: a blokk-egyenletességi feltétel mindig teljesül akkor, ha az osztályozandó pont csak valamelyik tanulópont lehet és mindegyik egyenlő valószínűséggel, illetve, ha mindig elég közel esik valamelyik tanulóponthoz.

Alakfelismerésben ennek teljesülése a jó lényegkiemelésen és távolságmértéken múlik. (Megjegyzendő, hogy eredményeink általánosabb feltételekkel jellemzett esetekre is kiterjeszthetők, de ezzel itt nem foglalkozunk.)

Feladatunk tehát megkonstruálni a T halmaz optimális dekompozícióját — amelyre a fenti várható érték minimális. Az összes lehetséges dekompozíció megvizsgálása számítástechnikailag kózbentarthatalan a tanulópontok számával exponenciálisnál is nagyobb mértékben növekvő műveletigénye miatt.

Az optimális dekompozíciót megkonstruáló polinomiális műveletigényű algoritmus az egyes tanulópontokat külön blokként tartalmazó dekompozícióból indul ki, és az egyes blokkokat egy képzési szabály szerint egyesítve több lépésben jut el az optimális dekompozícióig. A képzési szabály a következő:

Azt mondjuk, hogy a π^l dekompozíció *szabályszerűen képzett* π^m-ből, ha π^l összes blokkja egy kivételével n^m blokkja is, π^l fennmaradó blokkja pedig n^m fennmaradó k blokkjának egyesítése. k-nak olyannak kell lenni, hogy $m(\pi^l) \leqq m(\pi^m)$, és π^m nek k-nál kisebb számú blokkját ezzel a feltétellel nem lehet egyesíteni. Igazolható a következő tétel is ([10]), mely szerint ily módon valóban eljutunk az optimális dekompozícióig:

Legyen π_0 az egyes tanulópontokat külön blokként tartalmazó dekompozíció, π_{i+1} pedig szabályszerűen képzett π_i-ből ($i=1,2,\ldots,r$). Ekkor π_r az optimális dekompozíció, ha nincs π_{r+1}, amely ebből szabályszerűen képezhető.

Belátható, hogy minden dekompozícióra $m(\pi) \leqq 2\sqrt{N}-1$, és egyenlőség akkor áll fenn, ha az N tanulópontot $\sqrt{N}$ darab $\sqrt{N}$-elemű blokkra tudtuk particionálni.

Az algoritmus programozása

Annak ellenére, hogy a szabályszerű képzés definíciója bonyolultnak látszik, számítástechnikailag könnyen kezelhető. Az, hogy n elem önfüggő blokkot alkot-e, a rendezési tábla segítségével könnyen ellenőrizhető. A várható értékekre vonatkozó feltétel ellenőrzéséhez pedig belátható, hogy csupán az egyesítendő blokkok elemszámának vizsgálata szükséges, konkrétan az

$$(*) \qquad N(k-1) \geq \left[\sum_{i=1}^{k} n_i\right]^2 - \sum_{i=1}^{k} n_i^2$$

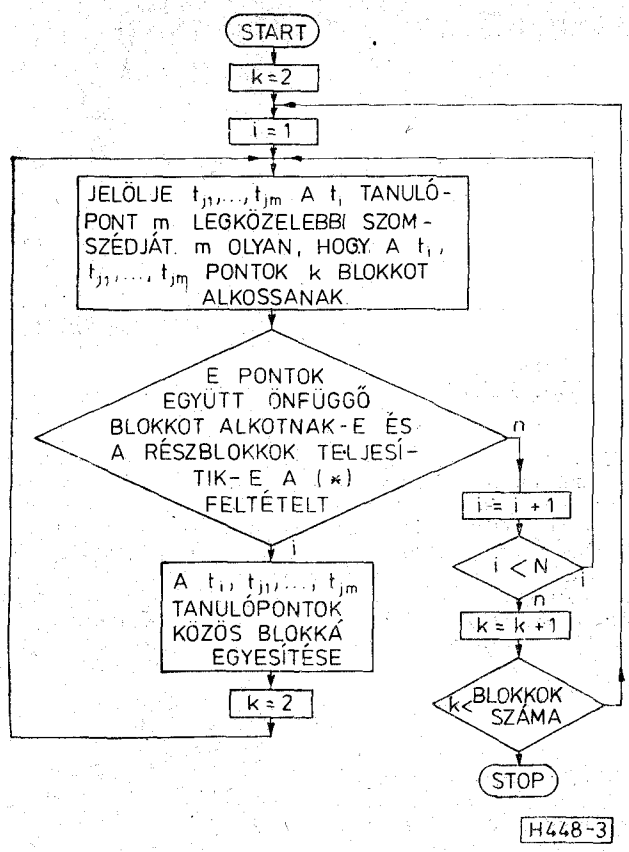
egyenlőtlenségnek kell teljesülnie. Az algoritmus folyamatábrája a 3. ábrán látható.

Hierarchikus dekompozíció

Miután eldöntöttük, hogy a legközelebbi szomszéd az előbb megadott dekompozíció melyik önfüggő blokkjának eleme, a keresést meggyorsíthatjuk az illető blokk további önfüggő blokkokra való particionálásával, majd a kapott blokkok további particionálásával stb., egészen addig, amíg a particionálással kapott blokkok maguk a tanulópontok nem lesznek. Világos, hogy így tovább csökkenthetjük a távolságszámítások számának várható értékét minden particionálási szinttel. Ennek ára a döntés hibavalószínűségének növekedése lehet.

Az eljárás lényegében hierarchikus clusterezés, azzal a módosítással, hogy a clusterek csak önfüggő blokkok lehetnek. Célunk a távolságszámítások számára optimális hierarehikus dekompozíció megkonstruálása tetszőleges tananyagra.

Minden egyes hierarchikus dekompozíció reprezentálható egy irányított fagráffal, melynek csúcsai a particionálás során kapott önfüggő blokkok. Egy csúcsból azokba a csúcsokba mutat él, amelyekre a particionálás során felosztottuk. Az összes



3. ábra. Az optimális dekompozíció előállítása, folyamatábra

tanulópontot tartalmazó blokk egy csúcsa a ráfnak, és belőle minden él kifelé mutat. Ugyanígy az egyes tanulópontok, mint blokkok szintén elemei a grófnak és egy-egy ól mutat feléjük, hiszen a többszintű döntés feladata az egyes tanulópontok azonosítása.

A hierarchikus dekompozíciót reprezentáló fagráf minden egyes csúcsához hozzárendeljük a gráf csúcsához tartozó önfüggő blokkon belüli tanulópontok megtalálásához szükséges távolságszámítások számának feltételes várható értékét azzal a feltétellel, hogy a keresett tanulópont az illető önfüggő blokk eleme. Így az egyes tanulópontokhoz rendelt feltételes várható érték 0 lesz, a T-hez, mint önfüggő blokkhoz rendelt feltételes várható érték pedig éppen a minimalizálandó várható érték.

Annak, hogy egy blokk reprezentáns pontja egy t_i pont legyen, egyedüli feltétele az, hogy a t_i eleme legyen a blokknak. Így, ha a K' önfüggő blokk reprezentáns pontja t_i, és a K' blokkot tovább partícionáljuk a $K_1, K_2, \ldots, K_r$ blokkokra, akkor ha $t_i \in K_j$, akkor t_i a K_j blokk reprezentáns pontja is lesz. Ha a reprezentáns pontokat nem így választanánk, az ehhez a választáshoz képest szintenként további egy távolságszámítást jelentene (kivéve a döntés első szintjét).

Az optimális hierarchikus dekompozíciót a következőképpen kaphatjuk meg (bizonyítása megtalálható [10]-ben): kiindulva az összes tanulópontot tartalmazó önfüggő blokkból azt a lehető legkevesebb számú önfüggő blokkra particionáljuk. Ezután az eljárást ugyanígy folytatjuk, minden lépés után a kapott önfüggő blokkokat a lehető legkevesebb számú önfüggő blokkra osztjuk, amíg meg nem kapjuk az összes tanulópontot egyenként, mint önfüggő blokkot.

E konstrukció azonban számítástechnikailag nehezen kezelhető, hiszen az egy blokkon belüli legnagyobb blokkokat kell megkeresni. Ezért helyette egy olyan ekvivalens (tehát az optimális hierarchikus dekompozíciót előállító) algoritmust adunk, amely jóval egyszerűbb: az egyes tanulópontokból kiindulva mindig a lehető legkevesebb számú blokkot egyesítjük, amíg a teljes tananyaghoz nem jutunk. Ez az eljárás a leállási feltételtől eltekintve lényegében azonos a 3. ábrán láthatóval, azzal a különbséggel, hogy itt nem vizsgáljuk a (*) feltételt. Így az optimális hierarchikus dekompozíciót is megkonstruálhatjuk polinomiális lépésszámú algoritmussal.

IRODALOM

[1] *P. E. Hart*, „Condensed nearest neighbor rule", IEEE Trans. Inform. Theory, vol. IT—14, pp. 515—516, May 1968.

[2] *C. L. Chang*, „Finding prototypes for nearest neighbor classifiers", IEEE Trans, Comput., vol. C—23, pp. 1179—1183, Nov. 1974.

[3] *Győrfi Z.*, „NN automaták", Tudományos Diákköri Dolgozat, 1974.

[4] *G. L. Ritter, H. R. Woodruff, S. R. Lowry, T. L. Isenhour*, „An algorithm for a seleetive nearest neighbor decision rule", IEEE Trans. Inform. Theory, vol. IT—21, pp. 665—669, Nov. 1975.

[5] *K. Fukunaga, P. M. Narenra*, „A branch and bound algorithm for computing k-nearest neighbors", IEEE Trans. Comput., vol. C—24, pp. 750—753. July 1975.

[6] *J. H. Friedman*, „An algorithm for finding nearest neighbors", IEEE Trans. Comput., vol. C—24, pp. 1000—1006, Oct. 1975.

[7] *T. P. Yunck*, „A technique to identify nearest neighbors", IEEE Trans, Syst., Man, Cybern, vol. SMC—6, pp. 678—683, Oct. 1976.

[8] *I. K. Sethi*, „A fast algorithm for reeognizing nearest neighbors", IEEE Trans. Syst., Man, Cybern., vol. SMC—11, pp. 245—248, Mar. 1981.

[9] *F. P. Fischer, E. A. Patrick*, „A preprocessing algorithm for nearest neighbor decision rules" in Proc. Nat. Electronics Conf., vol. 26, pp. 481—485, Dec. 1970.

[10] *Linder T., Lugosi G., Pikler T.*, „A legközelebbi szomszéd osztályozási módszer algoritmikus kérdései", Tudományos Diákköri Dolgozát, 1986 Október. (Konzulens: dr. Faragó András)

[11] *Faragó A., Linder T., Lugosi G., Pikler T.*, Publikálatlan munka

[12] *J. MacQueen*, „Some methods for classification and analysis of multivariate observations", Proc., 5th. Berkley Symp. on Math. Stat. and Prob., vol. 1, pp. 281—297, 1967.