Malliin pohjautuva hierarkkinen ryhmittely

Älykkäiden järjestelmien seminaari
15.03.2001
Jori Airamo

Tiivistelmä

Luon katsauksen malliin pohjautuvaan hierarkkiseen ryhmittelyyn, jonka lähtökohtana toimii Vaithyanathanin ja Domin artikkeli [17]. Tarkastelen, mistä tässä menetelmässä on kyse ja mitä ongelmia menetelmän mallin määrittämisessä ilmenee. Lisäksi esitän menetelmälle toteutettuja ratkaisumalleja ja niillä saavutettuja tuloksia.

1 Johdanto

Viime aikoina on osoitettu suurta kiinnostusta malliin pohjautuvaan ryhmittelyyn (model-based clustering). Malliin pohjautuvassa ryhmittelyssä data generoidaan käyttäen todennäköisyysjakaumien yhdistelmää (mixture of probability distributions), jossa kutakin komponenttia käsitellään ryppäänä. Malli sisältää kasautuvia (agglomerative) algoritmeja, jotka yhdistävät (merge) ryppäitä maksimoimalla tuloksen uskottavuuden (maximizing resulting likelihood).

Ongelmat. Mallin määrittämisen ongelmakenttiä ovat ryppäiden määrän valinta, ominaisuusjoukon (feature subset) valinta ja mallin parametrien määrän valinta. Näistä tutkituimman ongelman, ryväsmäärän valinnan, ratkaisuyrityksiä on esitetty esimerkiksi artikkeleissa [9] ja [15]. Banfield ja Raftery [1] kehittivät bayesilaisen lähestymistavan kyseiseen ongelmaan. Vaithyanathan ja Dom [18, 16] ovat tuoneet esille "unohtuneen" ongelman, joka käsittää ryppäiden muodostamisen ongelman eri ominaisuusavaruuksissa.

Vaithyanathan ja Dom. IBM:n tutkimuskeskuksen tutkijat Vaithyanathan ja Dom [18, 16] ovat tutkineet valvomatonta oppimista (unsupervised learning) ryhmityksessä ja luoneet bayesilaisen mallin 1-tasoiselle (flat) ryväsjoukolle, joka ratkaisi sekä ryväsmäärän että ominaisuusjoukon ongelman. Lisäksi he kehittivät MCCV-menetelmää (Monte Carlo cross-validation) käyttävän määrittelymallin. Edellisen mallin he laajensivat hierarkkiseksi artikkelissa [17]. Hierarkkisen mallin perusidea on se, että tietyillä ominaisuuksilla voi olla yhteinen jakauma joidenkin ryppäiden välillä samalla kun niillä on eri jakauma toisissa ryppäissä.

Algoritmit. Kasautuvia algoritmeja sisältävien mallien laajennoksia on tutkittu lähteessä [6]. Gaussilaisiin malleihin perustuvaa ryväsanalyysiä ja algoritmeja on tutkittu rohkaisevin tuloksin [10, 1, 6]. Lisäksi on kehitetty malliin pohjautuva HAC-algoritmi (hierarchical clustering algorithm), jonka toiminta perustuu multinomiseen yhdistettyyn malliin (multinomial mixture model) [11]. Vaithyanathan ja Dom ovat tutkineet erityisesti juuri multinomisia yhdistettyjä malleja ja he ovat laajentaneet HAC-algoritmin muotoon, joka yhdistää ryppäitä hierarkian kokonaisuskottavuuden (marginal likelihood of the hierarchy, MLH) kasvun mukaan [17].

Muita hierarkkisia ryhmitysalgoritmeja on lähteessä [8], jossa esitetään tekstikokoelmien ominaisuuksia (characterization) hierarkkisesti pohjautuen deterministiseen karkaisualgoritmiin (annealing algorithm). Kyseisessä mallissa käytetään latentteja muuttujia dokumenttien hierarkkiseen ryhmittelyyn ja välisolmujen määrittämiseen. Jälkimmäiset muuttujat, abstraktiosolmut (abstraction nodes), mallintavat sanojen ehdolliset todennäköisyydet. Regularisointi saavutetaan maksimoimalla erillisen testijoukon uskottavuus.

Vertailua. Artikkelissa [17] esitetty malli eroaa huomattavasti lähteessä [8] esitetystä työstä. Ensiksi vain osa ominaisuuksista mallinnetaan hierarkkisesti välisolmuissa (kyseisten ominaisuuksien valinta on osa mallin valitsemista). Tämä voi vähentää huomattavasti mallin parametrien määrää varsinkin suuren ulottuvuusmäärän omaavissa (high-dimensional) datajoukoissa. Toiseksi artikkelin [17] mallin regularisointi saavutetaan yhdistämällä parametriarvot Bayes-menetelmällä. Tämä mahdollistaa koko datajoukon käytön mallin rakentamisessa, eikä tällöin tarvita erillistä testijoukkoa. Lisäksi artikkelissa esitetty kaksivaiheinen algoritmi käyttää vain ensimmäisen vaiheen yhdistämislaskennassa saatuja riittäviä tilastoja (sufficient statistics), jolloin laskenta vähenee.

Ominaisuudet. Artikkelissa [17] kehitetty todennäköisyysmalli järjestää dataelementit luonnolliseen hierarkkiseen järjestykseen. Eri tasojen solmut määritetään mallintamalla *ominaisuudet* (features), jotka ovat yleisiä kyseessä olevan solmun lasten keskuudessa. Nämä yleiset ominaisuudet mallinnetaan käyttäen samaa jakaumaa kullekin ryppäälle. Kun ryppäillä on tällainen yleinen jakauma, niin jotkut ominaisuudet voivat olla hyödyllisiä muiden ryppäiden erottamisessa. Kyseisiä yleisiä ominaisuuksia kutsutaan solmun "kohinaksi" (noise) ja erottavia ominaisuuksia "hyödyksi" (useful). Tällainen analyysi johtaa luonnolliseen hierarkkiseen malliin, jota voidaan käyttää esimerkiksi suuresta dokumenttimäärästä tehtävään automaattiseen luokitteluun. Myöhemmin esitetään *kohinan* ja *hyödyn* formaalimpi määritelmä. Dokumenttien ryhmittelystä lisää esimerkiksi lähteissä [14] ja [5].

Sisältö. Seuraavaksi tarkastellaan kyseistä hierarkkista mallia ja todennäköisyyksiin perustuvaa toimintaa, jolla malli kuvataan. Kappaleessa 3 kuvataan joitakin skeemoja, joissa ratkaistaan mallin kuvauksen toiminta. Kappaleessa 4 tarkastellaan kokeellisia lähtökohtia ja tuloksia. Lopuksi kappaleessa 5 kerrotaan tulevaisuuden työstä.

2 Valvomattoman oppimisen mallit

Aikaisemmassa työssä [18, 16] artikkelin [17] kirjoittajat kuvailivat ongelmaa *1-tasoiseksi* (flat)[1], *osittavaksi* (partitional)[2] ryhmittelyksi. Tarkastellaan ensin tuota mallia yleisesti ja myöhemmin sen laajennosta eli hierarkkista *osittavaa* ryhmittelyä.

2.1 1-tasoisen osittavan ryhmittelyn malli

Edellisen mallin avainaspektina on ominaisuuksien osittaminen kahteen joukkoon (alijoukkoineen): "kohinaan" (N) ja "hyötyyn" (U). Määritellään kyseiset joukot seuraavasti.

kohinalla on sama tilastollinen jakauma kaikissa ryppäissä. Tällöin kohina on hyödytön ryppäiden erottamisessa.

hyödyllä on eri jakauma kussakin ryppäässä ja ne ovat hyödyllisiä ryppäiden erottamisessa.

Kirjoittajien edellisessä mallissa on vain nämä kaksi ominaisuustyypiä. Tällöin estyy mahdollisuus, että ominaisuuksilla olisi sama jakauma joissakin, mutta ei kaikissa ryppäissä. Nyt he lisäävät, että asia on näin vain tietyssä tapauksessa. Periaatteessa kahdella ominaisuudella voi olla identtiset tilastot kahdessa ryppäässä, jolloin niillä on myös identtiset parametriset arviot (estimates). Kyseisistä identtisistä arvoista huolimatta näitä kahta ryvästä käsitellään erillisinä

[1] 1-tasoinen ryvästys ei ole hierarkkinen

[2] Osittavassa ryvästämisessä kukin dataelementti kuuluu ainoastaan yhteen ryppääseen

ja siksi molemmilla on oma Bayesiläinen priorijakauma ja niitä käsitellään kahtena jakaumana regularisointitarkoituksiin. Tutkitaan tätä asiaa myöhemmin.

1-tasoinen ryvästysmalli (datan D todennäköisyydellä ehdolla *mallirakenteeseen* Ω) voidaan kirjoittaa yleisesti muotoon:

$$P(D|\Omega) = P(D^N|\Omega) \prod_{k \in \Psi} P_k(D_k^U|\Omega), \tag{1}$$

jossa:

D on datajoukon ominaisuusesitys.

Ω ilmaisee mallin emphrakenteen, joka ilmoitetaan seuraavasti: (1) ominaisuusjoukon osittaminen joukkoihin N ja U, (2) ryppäiden lukumäärä ja (3) yksittäisten dataelementtien $\{d_i\}$ määrittäminen ryppäisiin.

D^N ilmaisee datajoukon D projektiota *kohinan* aliavaruuteen N.

Ψ on ryväsjoukko indeksillä k.

D_k^U ilmaisee dataelementit ryppäässä k, D_k, projisoituna *hyödyn* aliavaruuteen U.

P_k on tietty todennäköisyysmalli liitettynä ryppääseen k.

Lähteissä [18] ja [16] Vaithyanathan ja Dom sovelsivat yleistä mallikaavaa 1 dokumenttien ryhmittelyn ongelmaan käyttäen *multinomijakaumia*. Bayesilainen[3] kokonaisuskottavuus (laskettiin käyttäen prioreina *Dirichlet-jakaumia*) oli kohdefunktio $P(D|\Omega)$. Käytetään samoja dokumenttien ryhmittelymalleja kuin kirjoittajat käyttivät lähteissä [18] ja [16].

2.2 Hierarkkisen ositusryhmittelyn malli

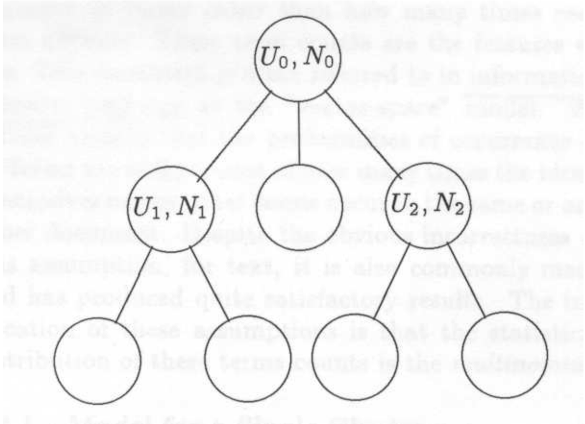
Laajennetaan 1-tasoinen ryhmittelymalli ryhmittelyhierarkiaksi (esitettynä puun avulla), jossa kaikki solmut vastaavat ryppäitä ja kaikki solmut paitsi lehdet jaetaan edelleen rekursiivisesti ryppäisiin, kunnes kaikki solmut ovat lehtiä. Kuvassa 1 on yksinkertainen esimerkki tällaisesta ryväshierarkiasta. Lähteessä [7] on keskusteltu samanlaisesta mallista kuin Vaithyanathanin ja Domin kehittämä malli artikkelissa [17].

Hierarkkisen laajennoksen tekemiseksi täytyy ensin laajentaa kaavan (1) notaatiota. Eri symbolit indeksoidaan käyttäen solmua, johon ne kuuluvat. Täten:

- $U, N \to U_k, N_k$, jotka ovat *hyöty* ja *kohina* liitettynä solmuun k. Näille tarvitaan lisäselvitystä, joka tehdään alla. Indeksiarvo $k = 0$ vastaa juurisolmua, joka sisältää datajoukon D (alias D_0).
- $\Psi \to \Psi_k$, solmun k *lapset*.

Artikkelin [17] kirjoittajien hierarkkinen malli laajentaa *kohina/hyöty*-käsitteen yleisempään tapaukseen kuin edellä esitetty. Yleisemmässä tapauksessa tietyt ominaisuudet voivat omata yhteisen jakauman, joka kattaa jonkun ryppäiden alijoukon. Kirjoittajien mallissa kaikki ryppäiden alijoukot eivät ole kandidaatteja tällaisille yhteisille ominaisuusjakaumille. Yhteinen ominaisuusjakauma liitetään johonkin solmuun ja sitä voidaan käyttää vain kyseiselle solmulle ja sen jälkeläisille eli *alipuulle*. Liitetyt ominaisuudet kuvataan symbolilla N_k. On huomattava, että jos solmu j on solmun k *esi-isä*, silloin $N_j \subseteq N_k$. Olkoon $p(k)$ k:n isäsolmu; määritetään:

[3] Bayesilaisen tilastollisen paradigman käsittely esitetään lähteessä [3]



Kuva 1: Yksinkertainen luokitteludiagrammi, johon on merkitty ominaisuusjoukkojen osat.

$$\tilde{N}_k \equiv N_k - N_{p(k)}. \tag{2}$$

N_k:n komplementtia eli solmun k ”hyöty”-ominaisuuksia merkitään muuttujalla U_k.

$$U_k + \tilde{N}_k = U_{p(k)}. \tag{3}$$

Ryhdytään rakentamaan kaavaan (1) pohjautuen hierarkkista mallia vaihtamalla ensin muuttujien merkintää:

$$P(D|\Omega) = P(D_0^{N_0}|\Omega) \prod_{k \in \Psi_0} P_k(D_k^{U_0}|\Omega). \tag{4}$$

Sitten laajennetaan termit $P_k(D_k^{U_0}|\Omega)$ rekursiivisesti:

$$P_k(D_k^{U_0}|\Omega) = P(D_k^{\tilde{N}_k}|\Omega) \prod_{l \in \Psi_k} P(D_l^{U_k}|\Omega). \tag{5}$$

Kullekin solmulle j, joka ei ole lehti, laajennetaan samoin:

$$P_k(D_k^{U_{p(k)}}|\Omega) = P(D_k^{\tilde{N}_k}|\Omega) \prod_{l \in \Psi_k} P(D_l^{U_k}|\Omega). \tag{6}$$

Valmis laajennettu muoto (6) on melko monimutkainen monimutkaisille ryväshierarkioille. Kaavaa käytetään rekursiivisesti, kunnes saavutaan puun lehtiin.

2.3 Hierarkian kokonaisuskottavuus

$P(D|\Omega)$:lle ja sen useille komponenteille (esimerkiksi $P(D_k^{U_{p(k)}}|\Omega)$ käytettäviä muotoja ovat *Bayesilaisia kokonaisuskottavuuksia* (Bayesian marginal likelihoods). Nämä perustuvat kahteen jakaumaan: varsinaiseen jakaumaan ja liitettyyn *priorijakaumaan.* Jakaumina voidaan käyttää esimerkiksi *multinomijakaumaa* ja multinomijakaumien tapauksessa Dirichletin priorijakaumaa.

Parametrijoukko ξ antaa kuvauksen useimmille jakaumille. Bayesilaisessa mallissa tätä joukkoa käsitellään satunnaismuuttujina, joilla on oma jakauma, priori, $\pi(\xi|\Omega)$. Ω ja ξ määräävät yksittäisen komponenttijakauman. *Kokonaisuskottavuus* muodostetaan integroimalla liitosjakauma $P(D, \xi|\Omega) = P(D|\xi, \Omega)\pi(\xi|\Omega)$ ξ:n suhteen:

$$P(D|\Omega) = \int P(D|\xi, \Omega)\pi(\xi|\Omega)d\xi.$$

Täten artikkelin [17] kirjoittajien mallissa kaikki termit kaavasta (1) kaavaan (6) laajennetaan tällä tavalla kappaleessa 2.2 esitetyn rekursiivisen hierarkkisen osituksen mukaisesti. Tämä laajennos on valmis, kun saavutaan puun lehtiin.

2.4 Multinomiperustainen hierarkkisen dokumenttiryhmittelyn malli

Jotta tätä yleistä lähestymistapaa voidaan käyttää dokumenttien hierarkkisen ryhmittelyn mallin rakentamiseen, aloitetaan käsittelemällä dokumenttia "pussillisella sanoja" eli jätetään huomioimatta kaikki muut tiedot paitsi termien frekvenssitiedot. Tiedonhakukielessä (information retrieval parlance) tällaista käsittelytapaa on usein viitattu vektoriavaruusmalliin (vector-space model). Edelleen oletetaan, että kaikkien termien tapahtumien todennäköisyydet ovat riippumattomia siitä, kuinka monta kertaa sama tai joku muu termi esiintyy missä tahansa dokumentissa. Tämän olettamuksen aiheuttamasta epätarkkuudesta huolimatta näin on yleisesti tehty käsiteltäessä tekstiä ja silti on päästy tyydyttäviin tuloksiin. Näiden olettamusten vaikutus on se, että termien frekvenssitietojen tilastollinen jakauma on multinominen.

2.4.1 Yksittäisen ryppään malli

Yksittäisen M-termisen sanaston sisältävän dokumentin d todennäköisyys termeillä $\{t_j|j = 1, 2, \dots, M\}$ on:

$$P(d|\xi) = \binom{n}{\{t_j\}} \prod_{j=1}^{M} \theta_j^{t_j}, \tag{7}$$

jossa $\binom{\cdots}{\{\cdots\}}$ on multinomikerroin, $n = \sum_{j=1}^{M} t_j$ ja θ_j on todennäköisyys sille, että sana on sanaston j:s termi. Yhdistetään joukko $\{\theta_j\}$ kappaleessa 2.3 olevan muuttujan ξ kanssa. Saman populaation (eli yksittäisen ryppään) kokoelman D, jossa on v kappaletta riippumattomia dokumentteja $\{d_i|i = 1, 2, \dots, v\}$, todennäköisyys kuvataan samaan tapaan kuin kaava (7):

$$P(d|\xi) = \prod_{i=1}^{v} \binom{n_i}{\{t_{i,j}\}} \prod_{j=1}^{M} \theta_j^{t_{i,j}}, \tag{8}$$

jossa kaavan (7) notaatiota on kasvatettu dokumentin indeksillä i.

Kokonaisuskottavuuden saamiseksi dokumenttikokoelmalle integroimme kaavan (8) tuloksen ja priorin joukon $\{\theta_j\}$ suhteen. Joukko $\{\theta_j\}$ on R^M:n simpleksi $\forall\theta_j \in [0, 1]$ ja normalisointirajalla (normalization constraint) $\sum_{j=1}^{M} \theta_j = 1$. Kuten yllä on esitetty, valitaan prioriksi *Dirichlet-jakauma* [3]:

$$\pi(\theta) = \frac{\Gamma(\alpha_0)}{\prod_{i=1}^{M} \Gamma(\alpha_i)} \prod_{j=1}^{M} \theta_j^{(\alpha_j - 1)}, \tag{9}$$

jossa $\alpha_0 = \sum_{j=1}^{M}$ ja $\{\alpha_i\}$ on *hyperparametrien* joukko. Tämän integroinnin tuloksena saadaan kaava, joka on *multinomisen Dirichlet-jakauman* (Multinomial-Dirichlet distribution) [2] termien tuotos. Kokonaisuskottavuus voidaan esittää seuraavasti:

$$P(D) = \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_0 + n)} \prod_{j=1}^{M} \frac{\Gamma(\alpha_j + t_j)}{\Gamma(\alpha_j)}, \tag{10}$$

jossa $t_j = \sum_{i=1}^{v} t_{i,j}$ ja $n = \sum_{j=1}^{M} t_j$. Dokumenttiryvästyksen kokeiluissa kappaleessa 4 α:t saavat kaavan (10) priorissa uniformin arvon 1.

2.4.2 1-tasoisen ryväsjoukon malli

Kun multinomisia ja Dirichlet-malleja käytetään ryväsjoukkoihin ominaisuusjoukolla (U, N), ositus tuottaa seuraavan kaavan [18]:

$$
\begin{aligned}
P(D|\Omega) = &\Big[\frac{\Gamma(\gamma_U+\gamma_N)\Gamma(\gamma_U+t^U)\Gamma(\gamma_N+t^N)}{\Gamma(\gamma_U)\Gamma(\gamma_N)\Gamma(\gamma_U+t^U+\gamma_N+t^N)}\Big] \\
&\times\Big[\frac{\Gamma(\beta_0)}{\Gamma(\beta_0+n^N)}\prod_{m=1}^{M_N}\frac{\Gamma(\beta_m+\tau_m)}{\Gamma(\beta_m)}\Big] \\
&\times\Big[\frac{\Gamma(\sigma)}{\Gamma(\sigma+v)}\prod_{k=1}^{K}\frac{\Gamma(\sigma_k+|D_k|)}{\Gamma(|D_k|)}\Big] \\
&\times\Big[\prod_{k=1}^{K}\frac{\Gamma(\alpha_0)}{\Gamma(\alpha_0+n_k^U)}\prod_{j=1}^{M_U}\frac{\Gamma(\alpha_j+\tau_{k,j})}{\Gamma(\alpha_j)}\Big]
\end{aligned}
\tag{11}
$$

jossa:

t^U ja t^N ovat täydelliset *hyöty*- ja *kohina*-termien frekvenssimäärät datajoukolle D mainitussa järjestyksessä.

τ_m on *kohina*-termien m määrä datajoukossa D.

$|D_k|$ on dokumenttien lukumäärä ryppäässä k.

$\tau_{k,j}$ on *hyöty*-termien j määrä ryppäässä k.

γ, β, σ ja α ovat hyperparametreja.

Kaavan (11) alkuperä selviää lähteestä [18]. Kaavassa (11) pätee seuraavat olettamukset: (1) kohina ja hyöty ovat ehdollisesti riippumattomia ja (2) kaikki parametrijoukot ovat riippumattomia.

2.4.3 Dokumenttijoukon hierarkkisen ryhmittelyn malli

Yhtälö (11) on esitys 1-tasoiselle ryväsjoukolle. Vaithyanathanin ja Domin [17] ryväshierarkiamalli (laajennettu kaavasta (4) kaavaan (6)) on huomattavasti monimutkaisempi. Heidän mallinsa sisältää kaavan (10) jokaiselle *kohinajakaumalle* - $P(D_0^{N_0}|\Omega)$ ja $\{P(D_k^{\tilde{N}_k}|\Omega)\}$ - ja yhden tällaisen kaavan kunkin lehden *hyötyjakaumalle* $l : P(D_l^{U_{p(l)}}|\Omega)$. Kuten yllä todettiin, kaikille parametriavaruuksille oletetaan muuttumattomat priorit.

3 Optimointialgoritmi

Optimointialgoritmin tehtävä on löytää yhtälön (4) maksimoiva mallirakenne $\hat{\Omega} \equiv argmax_\Omega P(D|\Omega)$, jossa joukko $\{P_k\}$ on rekursiivisesti laajennettu ja mallit - dokumenttien ryvästämisen tapauksessa multinomiset - vaihdetaan sovellukselle sopiviksi.

Tämä on erittäin vaikea tehtävä jopa multinomisissa tapauksessa, jossa integraaleille on suljettuja lauseita (closed-form expressions). Tämä johtuu siitä, että v kappaleelle olioita voi olla valtava määrä ryväshierarkioita. Käytetään kohtuullisen tehokasta lähestymistapaa, joka koostuu seuraavista kahdesta vaiheesta:

1. Esitä 1-tasoinen ryvästys, jossa ryppäiden määrä ja kohina-avaruus on määrätty optimointiyhtälön (11) avulla.

2. Muodosta hierarkia näistä ryppäistä käyttäen *MHAC-algoritmia* (modified hierarchical agglomerative clustering algorithm).

3.1 1-tasoisen ryvästyksen algoritmi

Esitetään 1-tasoisen ryvästyksen algoritmi käyttäen artikkeleissa [18] ja [16] kuvattua algoritmia. Algoritmi etsii maksimaalisen marginaalisen ratkaisun $\hat{\Omega}$ kaavalle (1) käyttäen kappaleen 2.4 malleja, EM-algoritmia ja *jakautuvaa ryhmittelyheuristiikkaa* (distributional-clustering heuristic) pienentäen ominaisuusosan (U, N) etsintäavaruutta. Bayesilainen *kokonaisuskottavuus* antaa regularisointikriteerin, joka mahdollistaa "luonnollisen" ryväsmäärän määrittämisen automaattisesti.

3.2 MHAC-algoritmi

Tässä 1-tasoisen ryvästyksen skeemassa datajoukon ositus on tehty alijoukkoihin N ja U. Ositus on osa optimaalista mallirakennetta $\hat{\Omega}$ kaavalle (1). Alijoukon N *kohinaominaisuudet* ovat globaaleja, sillä kaikilla ryppäillä on yhteinen jakauma. Hierarkiamalli ilmentää mallirakennetta Ω, jossa myös voi mahdollisesti olla yhteisiä jakaumia ryppäiden tietyillä alijoukoilla. Yhdistetyn hierarkian ja yhdistettyjen ominaisuusjoukkojen löytämiseksi käytetään muutettua versiota hierarkkisesta kasaavasta ryvästysalgoritmista, joka lyhennetään MHAC.

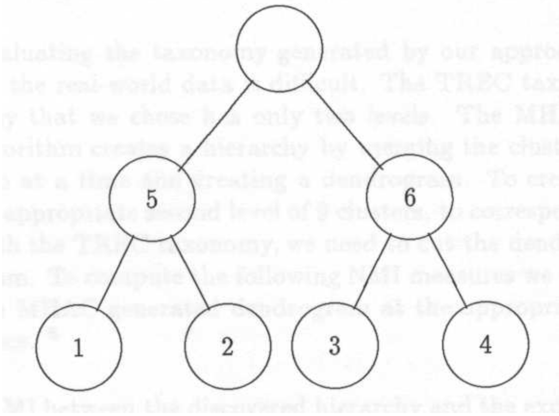
HAC-algoritmi aloittaa yksittäisistä ryppäistä (kukin datapiste on esitetty omana ryppäänään) ja on siksi laskennallisesti erittäin kallis algoritmi. Kuten lähteessä [12] on mainittu, dendrogrammin alemmat osat, jotka täten luodaan, eivät anna hyödyllistä tietoa. Tämän ongelman päihittämiseksi Posse [12] generoi ryppäiden alkujoukon käyttäen minimivirittävää puuta, jota muokattiin käyttäen heuristisia sarjoja. Muodostunutta tulosjoukkoa käytetään HAC-algoritmin syötteenä. Tämä lähestymistapa on erilainen kuin artikkelin [17] kirjoittajien, jossa ryppäiden alkujoukko luodaan käyttäen EM-algoritmia ja sitten ryppäät yhdistetään MHAC-algoritmilla.

MHAC-algoritmissa Vaithyanathan ja Dom [17] toteuttavat ryväsyhdentymiä, jotka kasvattavat kaavan (4) hierarkkista kokonaisuskottavuutta (marginal likelihood of the hierarchy, MLH). Ensin etsitään ominaisuuksien alijoukko, joka voidaan mallintaa kohinana kahdelle harkinnan alla olevalle ryppäälle. Näille ryppäille tehdään kohinan mallinnus yksittäisellä parametrijoukolla, jolloin kyseiset ryppäät kasvattavat MLH-arvoa. Algoritmi pysähtyy, kun mikään yhdentymistä ei enää kasvata MLH-arvoa. EM- ja MHAC-algoritmeja käyttävä algoritmi esitetään kokonaisuudessaan seuraavasti.

> Toista seuraavat kohdat, kunnes on vain yksi ryväs jäljellä tai MLH-arvo ei enää kasva
>
> > Etsi kaksi ryvästä, joiden yhdentäminen kasvattaa eniten MLH-arvoa
> >
> > > Laske ryppäistä i ja j MLH-arvon kasvu etsimällä kohinaominaisuudet
> >
> > Yhdistä edellä löydetyt ryppäät
> >
> > > Laske uudelle ryppäälle *riittävät tilastotiedot*[4] (summaa kahden yhdistetyn ryppään tilastotiedot)

Esitetään MHAC-algoritmissa käytettävän paikallisen kohinajoukon etsintä. Optimaalinen paikallisen alijoukon etsintä ominaisuusavaruudesta vaatisi $2^{N_{ij}}$ kombinaatiota kaikista ominaisuuksista. Koska tämä on laskennallisesti hankalaa, turvaudumme ahneeseen (greedy) algoritmiin, joka arvioi yhden ominaisuuden kerrallaan ja pysähtyy, kun uuden ominaisuuden lisäys ei lisää marginaalista todennäköisyyttä. Ensin ominaisuudet järjestetään muuttujan $\Delta\hat{\xi}^j$

[4] sufficient statistics



Kuva 2: I rakenne: Synteettiseen datakokeiluun käytetty ryväshierarkia.

mukaan nousevaan järjestykseen, jossa $\hat{\xi}^j$ on ominaisuuden j parametrin maksimitodennäköisyyden arvio ja $\Delta\hat{\xi}^j$ on muuttujan $\hat{\xi}^j$ ero kahden harkinnan alla olevien ryppäiden välillä[5]. Algoritmi lisää yhden ominaisuuden kerrallaan järjestetystä listasta, kunnes kokonaisuskottavuus alkaa laskea.

Katsotaan seuraavaksi MHAC-algoritmin näkökulmia. Aloitetaan MLH-arvon alkuarvolla, joka saadaan 1-tasoisen ryvästyksen algoritmilla[6]. MLH-arvoa ei lasketa uudelleen jokaisen yhdentymisoperaation jälkeen, sillä olemme kiinnostuneita vain MLH-arvon muutoksista.

MLH-arvon muutoksen laskemiseen käytetään teoreemaa (esitetty ja todistettu artikkelissa [17]), jonka mukaan kyseinen muutos on riippuvainen vain kahden ryppään riittävissä tilastotiedoissa (termin ryväskohtaisissa frekvenssitiedoissa) ja se voidaan laskea paikallisesti ilman ryppäissä olevaa varsinaista datajoukkoa (termin dokumenttikohtaisia frekvenssitietoja). Tätä faktaa on käytetty myös suurimman uskottavuuden toteutuksessa [11].

4 Kokeita

Kuvataan kokeiden suunnitelmia, joita käytetään arvioimaan Vaithyanathanin ja Domin [17] lähestymistapaa. Kokeissa käytetään sekä synteettistä että todellista dataa.

4.1 Datajoukot

4.1.1 Synteettinen data

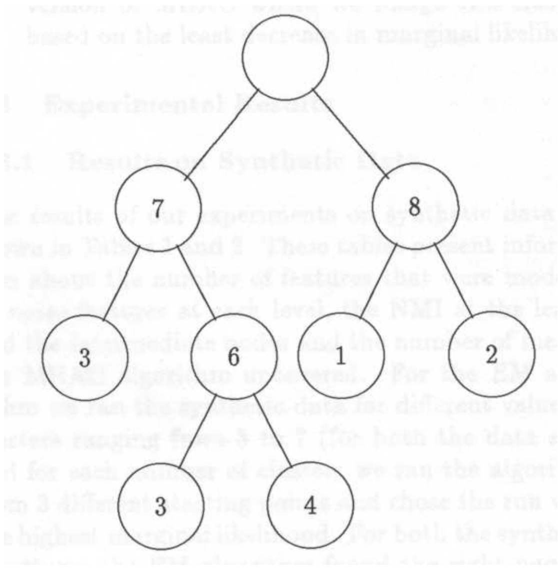
Artikkelissa [17] Vaithyanathan ja Dom generoivat kaksi synteettistä datajoukkoa perustuen olettamuksiin, jotka tehtiin mallin kehitys- ja tavoitteenasetteluvaiheessa. He määrittivät kaksi rakennetta (kuvat 2 ja 3) ja kummallekin rakenteelle he määrittivät *kohinan* kullekin välisolmulle parametrien kera, kuten kappaleessa 2 on kuvattu.

Sen lisäksi, että he generoivat mallin olettamuksiin sopivan datan, he asettivat seuraavat rajoitteet pienentämään kokeiden määrää.

- Kaikkiin datapisteisiin (olioihin) liitetään päätesolmu eli missään datapisteessä ei esiinny ainoastaan välisolmu.

[5]Heuristiikka perustuu siihen, että ominaisuudet, joilla on lähes samat kokonaisuskottavuusarviot molemmissa ryppäissä, olisivat parhaita kandidaatteja yksittäisen jakauman mallintamiseen

[6]Tämä hierarkia on verrattavissa puurakenteeseen, jossa juurisolmulla on yhtä monta lasta kuin 1-tasoisessa hierarkiassa on ryppäitä



Kuva 3: II rakenne: Synteettiseen datakokeiluun käytetty ryväshierarkia.

- Ominaisuuksien kokonaismäärä on 50.
- Välitasojen kohina ($\{\tilde{N}_k\}$ kaavoissa (2) ja (3)) generoitiin siten, että kunkin päätesolmun tai -ryppään hyötyjen kokonaistodennäköisyys oli vähintään 0,5.

Näitä todennäköisyyksiä noudattaen ja käyttäen tätä jakaumaa he testasivat kuvien 2 ja 3 rakenteita erilaisilla määrillä datapisteitä (taulukot 1 ja 2).

4.1.2 Todellinen data - TREC-dokumenttikokoelma

Artikkelin [17] kirjoittajat Käyttivät mallejaan ja algoritmejaan myös todelliseen dataan, joka koostui dokumenttikokoelmasta. Eräs tärkeä käytännön ongelma on järkevien luokittelujen muodostaminen suurista dokumenttikokoelmista. Heidän käyttämänsä dokumenttikokoelma oli AP Reuters Newswiren TREC-kokoelman artikkeleiden 50 ensimmäistä aihetta. Kukin aiheista, jotka muodostavat luokittelun lehtisolmut, kuuluu johonkin yhdeksästä alueesta (domain) - ja nämä alueet muodostavat luokittelun seuraavaksi korkeimman tason.

4.2 Kokeiden kuvaus

Parhaan $\hat{\Omega}$:n arvion löytämiseksi käytettiin MHAC-algoritmia kaikille datajoukoille. Ensimmäisessä vaiheessa käytettiin EM-algoritmin sovellusta[7]. Molemmat datarakenteet ryhmiteltiin eri ryväsmääriin kolmen ja seitsemän ryppään välillä. Kullekin näille ryväsmäärille ajettiin EM-algoritmi kolmella eri siemenluvulla (siemenluvut generoitiin artikkelissa [11] olevan metodin avulla). Tämän jälkeen ryppäiden optimaalinen lukumäärä saadaan ryvästämällä korkeimman kokonaisuskottavuuden mukaan. Näitä ryppäitä käytetään sitten MHAC-algoritmissa ja hierarkia rakennetaan kulkien alhaalta ylös.

Näissä kokeissa ollaan kiinnostuneita arvioimaan algoritmin kykyä seuraavissa tehtävissä

1. Selvitä oikea lehtiryppäiden lukumäärä ja jaa data ryppäisiin oikein.
2. Paljasta määrätty hierarkkinen rakenne.

[7]on huomattava, että juurisolmulle ei generoitu kohinaa (siis $N_0 = 0$) estäen kohinan etsinnän tällä tasolla

size	MHAC Node 5 actual (found)	MHAC Node 6 actual (found)	NMI (leaf)	NMI (inter)
25000	15(0)	21(7)	0.7950	NA
50000	15(4)	21(7)	0.795	1.00
75000	15(3)	21(9)	0.7998	0.99
100000	15(4)	21(9)	0.7967	1.00

Taulukko 1: Kuvan 2 rakenteen synteettisen datan tulokset.

3. Löydä oikeat kohinat hierarkkisen rakenteen kullakin tasolla.
4. Laske normalisoidut yhteiset tiedot (normalized mutual information, NMI) [18] löydetyn ja alkuperäisen hierarkian välillä sekä lehdissä että välitasolla. NMI=1 tarkoittaa luokka- ja ryväslabeleiden täydellistä yhteisymmärrystä ja NMI=0 tarkoittaa, ettei korrelaatiota ole lainkaan.

Todellisen datan luokittelun arviointi kyseisellä lähestymistavalla on vaikeaa. Kirjoittajien valitsemalla TREC-luokittelulla on vain kaksi tasoa. MHAC-algoritmi luo hierarkian yhdistämällä kaksi ryvästä kerrallaan tehden samalla dendrogrammin. Dendrogrammi pitää katkaista, jotta voidaan luoda edellä mainitun yhdeksän ryppään kokoinen toinen taso. Seuraavien NMI-mittojen laskemiseksi katkaistaan MHAC:n generoima dendrogrammi oikeasta kohdasta (MHAC-algoritmi siis yhdistää ryppäitä).

Löydetyn ja alkuperäisen hierarkian välinen NMI-arvo molemmilla tasoilla.

Löydetyn ja alkuperäisen hierarkian välinen NMI-arvo molemmilla tasoilla käyttäen yksinkertaista MHAC-algoritmin versiota, jossa kaksi ryvästä yhdistetään perustuen kokonaisuskottavuuden pienimpään alenemiseen.

4.3 Testitulokset

4.3.1 Synteettisen datan tulokset

Synteettisen datan testitulokset ovat taulukoissa 1 ja 2. Taulukoissa esitetään kohinaksi mallinnettujen ominaisuuksien lukumäärä kullakin tasolla, NMI-arvo lehdissä ja välisolmuissa sekä MHAC-algoritmin paljastama yhdentymien määrä. Artikkelin [17] kirjoittajat ajoivat molemmille datajoukoille EM-algoritmin 3–7 ryppään määrillä ja kullekin ryväsmäärälle he ajoivat algoritmin käyttäen kolmea eri alkupistettä. Lisäksi he valitsivat korkeimman mahdollisen kokonaisuskottavuuden. EM-algoritmi löysi molemmille rakenteille oikean ryväsmäärän. Lehden NMI-arvo on todellisen datan ryväsjaon ja EM-algoritmin löytämän ryväsjaon välinen normalisoitu yhteinen tieto. Tätä arvoa kuvaa NMI (leaf) taulukossa 1.

Lehden NMI-arvo lasketaan seuraavasti käyttäen apuna datajoukkoa 1. MHAC-algoritmi käytti kaikkia neljää lehtiryvästä yhdistämiseen ja välisolmujen etsimiseen. Kun MHAC-algoritmi jatkaa yhdistämistä niin kauan kuin MHAC kasvaa tai yhdistämisiä ei enää voida tehdä, taulukosta 1 käy ilmi, että MHAC-algoritmi sai aikaan vain yhden yhdistymisen 25000 testikoolla. Muut testikoot tuottivat maksimissaan kaksi yhdentymistä. Kahden yhdentymisen jälkeen lasketaan löydettyjen väliryppäiden ja alkuperäisen kahteen väliryppääseen tehdyn datajaon välisen NMI-arvon (NMI (inter) taulukossa 1). Samanlainen analyysi datajoukolle 2 esitetään taulukossa 2, jossa tuotettiin maksimissaan kolme yhdentymistä.

size	MHAC Node 6 actual (found)	MHAC Node 7 actual (found)	MHAC Node 8 actual (found)	NMI (leaf)	NMI (inter)
25000	24(0)	20(0)	15(0)	0.94	0.63
50000	24(13)	20(8)	15(6)	0.94	1.00
75000	24(16)	20(9)	15(6)	0.94	1.00
100000	24(14)	20(9)	15(6)	0.94	1.00

Taulukko 2: Kuvan 3 rakenteen synteettisen datan tulokset.

Level	NMI MHAC-FS	NMI MHAC-noFS
Level 2	0.513	0.513
Level 1	0.300	0.280

Taulukko 3: Todellisen dokumenttikokoelman tulokset.

4.3.2 Todellisen dokumenttikokoelman tulokset

Vaithyanathan ja Dom tekivät artikelissa [17] todellisen maailman datajoukolle ominaisuuksien laajennoksen. Kyseinen menetelmä on kuvattu lähteissä [4] ja [13]. Ominaisuuksien valinta suoritettiin käyttäen lähteessä [18] olevaa jakautuvaa ryhmittelyalgoritmia (distributional clustering algorithm). Ominaisuuksien valinnan jälkeen jäi 14772 tokenia. Jotta voidaan saavuttaa oikeanlainen vertailu TREC-luokittelulle, kaikista 16801 dokumentista muodostettiin 50 ryvästä kolmella eri alkupisteellä (alkuosituksella) ja valittiin ajo, joka tuotti korkeimman kokonaisuskottavuuden. Nämä 50 ryvästä yhdistettiin käyttäen kahta MHAC-algoritmin versiota, toisessa on ominaisuuden valinta (feature selection, MHAC-FS) ja toisessa ei (MHAC-noFS). Versio, jossa ei ole ominaisuuden valintaa, yksinkertaisesti yhdisti ryppäitä perustuen kokonaisuskottavuuden pienimpään alenemiseen. Testien tuloksista (taulukko 3) keskustellaan seuraavassa alikappaleessa.

4.4 Keskustelu

Kuten taulukoista 1 ja 2 nähdään, EM+MHAC-algoritmi löysi rakenteen paljon helpommin ensimmäisellä datajoukolla kuin toisella. EM-algoritmi löysi molemmissa tapauksissa oikean ryväsmäärän kullakin testikoolla. Tämä voidaan osittain selittää sillä, että kunkin lehtisolmun hyötyjen todennäköisyys pakotettiin olemaan vähintään 0,5. Jos hyötyjen määrä on pieni lehtitasolla, EM-algoritmin voi olla vaikea saada oikeaa lehtisolmujen määrää ja voidaan joutua turvautumaan muihin arvioihin. Tässä kokeessa ei kuitenkaan jouduttu harjoittamaan mitään muita arvioita.

NMI-arvo on suurempi välitasoilla, joka kertoo siitä, että EM-algoritmin suorittaman jaon sekaannuksen lehtisolmuissa saattoi aiheuttaa välitasojen kohina. Toinen tärkeä tekijä on se, että toisen rakenteen toisen tason NMI-arvo on täydellinen niille testeille, joissa testikoko oli suurempi kuin 25000. Toisin kuin ensimmäisessä rakenteessa, jossa MHAC-FS -algoritmi löysi vain yksittäisen yhdistymisen (muut yhdistymiset eivät nostaneet MLH-arvoa), toisessa rakenteessa algoritmi teki vääriä yhdistymisiä testikoolla 25000. Tämä vaikuttaa jyrkästi kyseisen datajoukon NMI-arvon laskuun (katso taulukko 2).

Todellisen datajoukon testissä voidaan havaita, että MHAC-FS -algoritmi parantaa hieman MHAC-noFS -algoritmin tehoa. MHAC-FS -algoritmin lisäetuna on se, että malli tarjoaa myös listan kunkin välisolmun yhteisistä ominaisuuksista. Näitä ominaisuuksia voidaan käyttää luokittelun sovelluksessa labeleina kuvaamaan välisolmuja.

Vaithyanathan ja Dom [17] tekivät myös pienimuotoisen käyttäjätutkimuksen, jossa käyttäjille annettiin 20 satunnaisesti valittua dokumenttia alkuperäisten 16801 joukosta. Yksi käyttäjäjoukko sai tehtäväkseen luokitella nämä dokumentit EM+MHAC-algoritmin (täysin automaattisesti) generoimaan hierarkiaan. Toinen käyttäjäjoukko sai tehtäväkseen luokitella samat dokumentit alkuperäisen hierarkian (TREC-hierarkian) mukaan. Käyttäjien suoritukset arvosteltiin väärien luokittelujen ja käytetyn ajan mukaan. Käyttäjältä kului EM+MHAC-algoritmin generoiman hierarkian luokitteluun keskimäärin 15 minuuttia enemmän ja 7,5% huonommalla tarkkuudella. 20 dokumentin luokitteluun TREC-hierarkian mukaan kului kokonaisuudessaan 45 minuuttia.

5 Yhteenveto ja tulevaisuuden työ

Vaithyanathan ja Dom [17] ovat esittäneet uuden mallin, jolla olioita voidaan järjestää luonnolliseen hierarkiaan. Pääidea on siinä, että tietyillä ominaisuuksilla voi olla yhteinen jakauma toisten ryppäiden kanssa ja eri jakauma toisten kanssa. Tälle mallille on tehty useita sovelluksia mukaan lukien automaattisen luokittelun generoinnin ongelma. Mallin päätoiminta perustuu bayesilaiseen analyysiin ja Vaithyanathan ja Dom ovat johtaneet hierarkian bayesilaiselle kokonaisuskottavuudelle mallin tietyillä olettamuksilla. He tarjoavat uuden algoritmin, joka toimii kahdessa vaiheessa: ensin generoidaan 1-tasoinen ositusjoukko, jonka jälkeen käytetään HAC-algoritmin paranneltua versiota (MHAC) mallintamaan välisolmut ja ominaisuudet, jotka ovat yhteisiä kunkin välisolmun lapsille. He käyttivät tätä mallia kahden synteettisen ja yhden todellisen datajoukon kanssa.

Rohkaisevista tuloksista huolimatta useita tutkimusmahdollisuuksia jäi avoimeksi koskien mallia ja algoritmeja. Hierarkkisen mallin laajennuksia ja monimutkaisempia rakenteita kuten DAG (directed acyclic graphs) tulisi tutkia. Lisäksi Vaithyanathan ja Dom käyttivät kaikissa testeissä muuttumatonta prioria $\pi(\xi)$ - muita prioreja tulisi tutkia. Algoritmissa on useita kohtia, jotka pitää nostaa esille. Esimerkiksi on tarve päästä eroon MHAC-algoritmin rajoitteesta yhdistää vain kaksi ryvästä - siis kohina tulisi mallintaa lapsiryhmien (>2) kesken. Kirjoittajat suunnittelevat tutkimusta näille laajennuksille tulevaisuudessa.

Viitteet

[1] J. Banfield and Adrian Raftery. Hierarchical model-based clustering for large datasets. *Biometrics*, 49:803–821, 1993.

[2] J.M. Bernardo. Reference posterior distributions for bayesian inference. *Journal of the Royal Statistical Society*, 41:113–147, 1979.

[3] J.M. Bernardo and A.F.M. Smith. *Bayesian Theory*. Wiley, New York, 1994. ISBN 0-471-92416-4.

[4] Roy Byrd, Yael Ravin and John Prager. Lexical assistance at the information retrieval user interface. In *Proceedings of the Symposium on Document Analysis and Information Retrieval*, 1995.

[5] Douglas Cutting, David Karger, Jan Pederson and John Tukey. Scatter/gather: a cluster-based approach to browsing large document collections. In *SIGIR*, 1992.

[6] Chris Fraley. Algorithms for model-based gaussian hierarchical clustering. *SIAM Journal on Scientific Computing*, 1993.

[7] Robin Hanson, John Stutz and Peter Cheeseman. Bayesian classification with correlation and inheritance. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 692–698, San Francisco, CA, 1991. Morgan Kaufman.

[8] Thomas Hofmann. Cluster-abstraction model: unsupervised learning of topic hierarchies from text data. In *International Joint Conference on Artificial Intelligence*, 1999.

[9] Kontkanen et al. Comparing Bayesian Model Class Selection Criteria by Discrete Finite Mixtures, ISIS'96, pages 367–374, 1996.

[10] G. McLachlan and K. Basford. *Mixture Models: Inference and Applications to Clustering.*, Marcel Dekker, New York, 1988.

[11] Marina Meila and David Heckerman. An experimental comparison of several clustering and initialization methods. Technical Report MSR-TR-98-06, Microsoft Research Division, 1998.

[12] Christian Posse. Hierarchical model-based clustering for large datasets. Technical Report TR363, University of Washington, Seattle, Department of Statistics, 1999.

[13] Yael Ravin and Nina Wacholder. Extracting names from natural-language text. Technical Report RC-20338, IBM Research Division, 1996.

[14] Meheran Sahami. Using machine learning to improve information access. Technical report, Department of Computer Science, Stanford University, 1998.

[15] P. Smyth. Clustering using Monte Carlo cross-validation, KDD, 1996.

[16] Shivakumar Vaithyanathan and Byron Dom. Generalized model selection for unsupervised learning in high dimensions. In S. A. Solla, T. K. Leen and K. R. Müller, editors, *Proceedings of Neural Information Processing Systems*. MIT Press, November 1999.

[17] Shivakumar Vaithyanathan and Byron Dom. Model-based hierarchical clustering. In *Sixteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-2000)*, 2000.

[18] Shivakumar Vaithyanathan and Byron Dom. Model selection in unsupervised learning with applications to document clustering. In I. Brakto and S. Dzeroski, editors, *Proceedings of the Sixteenth International Conference on Machine Learning*, pages 423–433, San Francisco, June 1999. Morgan Kaufman.