Kümeleyici Topluluklarının Başarısını Etkileyen Faktörler
The Performance Factors of Clustering Ensembles

M. Fatih Amasyalı[1], Okan Ersoy[2]

1. Bilgisayar Mühendisliği Bölümü
Yıldız Teknik Üniversitesi
mfatih@ce.yildiz.edu.tr

2. Elektrik ve Bilgisayar Mühendisliği Bölümü
Purdue Üniversitesi
ersoyC@purdue.edu

Özetçe

Sınıflandırıcıların kararlarının birleştirilmesinde elde edilen başarılar kümeleme sonuçlarının birleştirilmesi alanında çalışmalara kaynak oluşturmuştur Bu çalışmada kümeleyici topluluklarının performanslarını etkileyen çeşitli faktörler (kümeleme algoritmasını türü, kümeleme algoritmalarında kullanılan özellik sayısı, kümeleyici sayısı, kümeleyici kararlarının birleştirilmesinde kullanılan algoritma) 15 veri kümesi üzerinde incelenmiş ve birbirlerine göre karşılaştırmaları yapılmıştır. Kümeleyicilerin veri kümesinin farklı özellik alt kümeleri üzerinde verdikleri kararlar birleştirilmiştir. Karar birleştirmede ortalama olarak graf tabanlı algoritmalar başarılı olurken, en yüksek bireysel başarıları hiyerarşik algoritmalar göstermiştir. Kararları birleştirilen kümeleyicilerin kullandıkları özellik sayısının artışı, kümeleyici sayısının artışı performansı arttırmıştır. Kümeleme algoritmalarında kmeans ve fuzzy kmeans en başarılı kümeleme algoritmaları olmuştur.

Abstract

The accomplishments on classifier ensembles originate the studies of clustering ensembles. In this study the factors on performance of clustering ensembles (clustering algorithm, the number of features used in clustering, the size of ensemble, the decision combining algorithm) are investigated and compared on 15 benchmark datasets. The decisions of clustering algorithms based on different feature subsets are combined. On the process of decision combination, the graph partition algorithms are averaged successful while hierarchical algorithms have best individual successes. The number of features used in clustering algorithms increases the success. The size of clustering ensemble is also direct proportional with clustering performance. Kmeans and fuzzy-kmeans are best clustering algorithms over our experimented datasets.

1. Giriş

Kümeleme, veri kümesi içindeki benzer örnekleri içeren kümelerin ve bu kümeleri temsil eden merkezlerin bulunması olarak tanımlanabilir. Bir küme içindeki örnek kendisiyle aynı küme içindeki örneğe, bir başka küme içindeki örneğe göre daha çok benzemektedir. Kümeleme sayesinde verilerin içerdikleri farklı tür sayısı azaltılarak veriler sıkıştırılabilir ya da verinin yapısı ortaya çıkarılabilir. Büyük miktarda örnek içeren ya da fazla sayıda özeliğe sahip veri kümelerinde bu işlemin elle yapılması mümkün değildir. Bu ihtiyaçtan dolayı bilgisayarlarda uygulanmaları için birçok kümeleme algoritması geliştirilmiştir. Özellikle son zamanlarda sınıflandırıcı kararlarının birleştirilmesiyle elde edilen yüksek performanslar kümeleme alanında da ilgi çekmiş ve tek bir kümeleme algoritması kullanmak yerine bir kümeleyici topluluğu oluşturarak onların ortak kararlarının kullanılması gündeme gelmiştir.

Bu çalışmamızda kümeleyici topluluklarının performansı üzerinde etkin faktörler araştırılmıştır. Kümeleyici topluluğunun oluşturulma yöntemi, kümelemede kullanılan algoritma, kümeleyicilerin kullandıkları özellik sayısı ve kümeleyici sayısı bu faktörler arasında yer almaktadır.

2. Kümeleme Metotları

Bu bölümde kararları birleştirilecek olan kümeleme metotları kısaca anlatılacaktır. Kmeans, Fuzzy K-means, SOM ve Eklemeli kümeleme literatürde en çok kullanılan kümeleme algoritmaları olduklarından seçilmişlerdir. Bu bölümün devamında bu algoritmalar kısaca anlatılmıştır [2].

2.1. K-Means

En basit ve en sık kullanılan kümeleme algoritmalarındandır. Algoritma rasgele seçilen küme sayısı (K) adet merkez noktayla başlar. Veri kümesindeki her nokta kendisine en yakın merkez noktasının kümesine atanır. Küme merkezinin değeri kendine ait noktaların ortalaması alınarak hesaplanır. Bu işlem küme merkezlerinin değerleri değişmeyinceye kadar devam eder.

2.2. Fuzzy K-Means

Lutfi Zadeh tarafından önerilen Fuzzy kümeler teorisinin Bezdek[1] tarafından K-Means'e uygulanmasıdır. Kmeans'de bir örnek sadece kendisine en yakın merkezin kümesine aittir. Fuzzy versiyonda ise her örnek her merkeze ona olan uzaklığıyla ters orantılı bir oranla aittir.

2.3. Self Organizing Maps (SOM)

Kohonen tarafından önerilmiştir [5]. Algoritma küme merkezlerinin birbirlerine bir ızgara üzerinde bağlı olduğunu kabul eder ve kazanan merkezin (girişi en yakın olan) hem kendisinin hem de komşuları örneğe belli bir oranda yaklaştırılarak güncellenir.

2.4. Hiyerarşik Kümeleme

Örneklerin birbirlerine yakınlıklarına göre hiyerarşisini çıkaran algoritmadır. Temelde iki yaklaşıma sahiptir. İlki başlangıçta örneklerin her birini bir küme olarak kabul edip her bir adımında kümelerin birbirlerine uzaklığını hesaplayıp o anda birbirine en yakın olan iki kümeyi birleştiren algoritmadır. İkincisi ise başlangıçta tüm örnekleri tek bir kümeye koyup her seferinde birbirine en uzak iki küme bulmaya çalışan algoritmadır. Her iki işlem sırasında hiyerarşik bir ağaç ortaya çıkar. Diğer algoritmalardan farklı olarak örneklerin koordinatlarına ihtiyaç duymaz. Örneklerin arasındaki mesafe matrisi bu algoritma için yeterlidir. Bu sebeple kümeleme kararlarını birleştirilmesi kısmında da literatürde kullanılmıştır [7].

3. Kümeleme Topluluklarının Üretilmesi ve Kararlarının Birleştirilmesi

Sınıflandırma başarısı %X olan (X>50) birden fazla ikili sınıflandırıcının kararları birleştirildiğinde sınıflandırıcı kararları birbirinden bağımsız iseler başarıları X'den daha büyük olur [2]. Çünkü yanlış karar vermeleri için sınıflandırıcıların yarıdan fazlasının yanlış karar vermiş olmaları gerekir. Sınıflandırıcı kararlarının birbirinden bağımsızlığı eğitildikleri veri kümelerinin birbirlerinden bağımsızlığıyla doğru orantılıdır. Sınıflandırıcılar için geçerli olan bu metot kümeleyiciler içinde uygulanmaktadır.

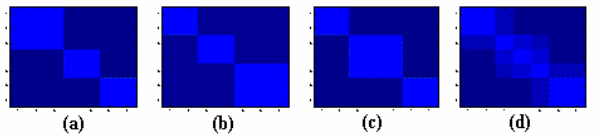
Kümeleyici kararlarının ve kümelenen data setlerin birbirlerinden bağımsız olmasını sağlamak için çeşitli metotlar literatürde önerilmiştir. Derek ve arkadaşları [6] çalışmasında 7 farklı komite üretme metodu denemiş ve ürettikleri data setlerin birbirlerinden bağımsızlığı en fazla olanın rasgele örnek alt uzayı (random subspacing) olduğunu göstermiştir. Bu nedenle bu çalışmada subspacing olarak adlandırılan, örneklerin rasgele seçilen özellik alt kümeleriyle çeşitli veri kümelerinin oluşturulması metodu kullanılmıştır.

Her biri orijinal data setin bir alt kümesine göre karar veren kümeleyicilerin kararlarının birleştirilmesi için literatürde temelde graf bölümleme ve hiyerarşik olmak üzere 2 farklı metot önerilmiştir.

3.1. Graf Bölümleme Metotları

Graf bölümleme algoritmalarında giriş, köşeler ve ağırlıklandırılmış kenarlardan oluşur [7]. Amaç minimum sayıda ve ağırlıktaki kenarı keserek, grafı K adet eşit büyüklükteki parçaya bölmektir. Kümeleyici kararlarının graf bölümleme algoritmalarıyla çözümlenebilmesi için kümeleme sonuçlarının graf formatına çevrilmesi gerekir. Strehl ve Ghosh [8] graf bölümlemeyi kümeleyici topluluklarında kullanımına dair örnek tabanlı ve küme tabanlı iki yaklaşım önermişlerdir.

Örnek tabanlı yaklaşım: Grafın köşeleri örnekler, kenar ağırlıkları kenarın bağladığı iki köşedeki örneklerin kaç kümeleme sonucunda aynı küme içinde yer aldıklarıdır. Şekil 1 (a,b,c)'de 7 örneğin 3 kümeleme sonucunda birlikte yer almaları verilmiştir. Şekil 1(d)'de ise ortalama olarak kaç kümeleme sonucunda aynı kümede kaldıkları verilmiştir. Graf bölümleme algoritmasına son benzerlik matris verilir [7] ve Eşitlik 1'deki şekilde bulunur.



Şekil 1: 3 farklı kümeleme sonucunun birleştirilmesi. Renklerin açıklığı ilişkinin gücüyle doğru orantılıdır

$$W(i,j) = \frac{1}{K}\sum_{t=1}^{K} I(g_t(X_i) = g_r(X_j)) \qquad (1)$$

Eşitlik 1'de I(.) içindeki ifade doğru olduğunda 1, olmadığında 0 döndüren bir fonksiyondur. $g_r(.)$ bir örneğin ait olduğu kümeyi döndürür.

Küme tabanlı yaklaşım: Grafın köşeleri kümeler, kenar ağırlıkları kenarın bağladığı iki köşedeki kümenin birbirlerine Jaccard Ölçümüne göre benzerliğidir ve Eşitlik 2'deki şekilde bulunur.

$$W(i,j) = \frac{\left|C_i \cap C_j\right|}{\left|C_i \cup C_j\right|} \qquad (2)$$

Ortak eleman sayılarının, birleşim kümesinin eleman sayısına oranı Jaccard ölçümü olarak tanımlanmıştır. Birçok kümeleyici sonucundan, kümeler arası benzerlik matrisi elde edildikten sonra kümelerde en fazla birlikte geçen örnekler aynı kümelerde kabul edilerek sonuç kümeleme elde edilir.

Strehl ve Gosh'un makalesinde örnek tabanlı, küme tabanlı ve ayrıntılarına burada girmeyeceğimiz meta küme tabanlı 3 metotla (cspa, hgpa, mcla) yaptıkları denemeler verilmiştir [8]. Bu üç metottan en iyi sonucu vereni Normalized Mutual information ile seçen bir uygulama geliştirmişler ve kullanıma sunmuşlardır. Bu çalışmada hiyerarşik algoritmalarla bu uygulamanın sonuçları karşılaştırılmıştır.

3.2. Hiyerarşik (agglomerative) metotlar

Graf bölümleme kısmında örnek tabanlı algoritmada sözü edilen, örneklerin kaç adet kümeleme sonucunda aynı küme içinde yer aldıklarını gösteren benzerlik matrisi bir çeşit örnekler arası mesafenin matrisi olarak da düşünülebilir. Bu durumda, elde örnekler arası mesafeler varsa her adımda birbirine en benzer iki kümeyi birleştirerek kümeleme yapan hiyerarşik kümeleme metotları kullanılabilir.

Hiyerarşik metotlarda iki kümenin birbirine benzerliğinin bulunmasında complete, average, weighted, centroid, median ve ward olmak üzere Matlab [4] yazılımında yer alan 6 farklı metot denenmiştir.

Bu metotlardan hangisinin kullanılacağına 11 farklı data set üzerinde yapılan denemelerle karar verilmiştir. Tablo 1'de 6 farklı metodun 11 veri kümesinde sınıf sayısı kadar küme sayısı bulan K-Means sonuçlarını birleştirdiğindeki performansları görülmektedir.

Tablo 1: İki kümenin benzerliğini ölçen metotların karşılaştırılması

datasetler	v1	v2	v3	v4	v5	v6	v7	v8	v9	v10	v11	Ort.
Complete	.86	.85	.89	**.71**	.81	.59	**.84**	.38	.81	.68	.67	.74
Average	.86	.83	.87	.66	.82	.6	.83	.38	.79	**.69**	.68	.73
weighted	.86	.85	**.89**	.7	.81	**.62**	.83	**.4**	.7	.67	.68	.73
Centroid	**.89**	.84	.86	.7	**.85**	.59	.83	.4	**.84**	.68	**.68**	.74
Median	.86	.86	.88	.69	.82	.56	.83	.39	.78	**.69**	**.68**	.73
Ward	.84	**.88**	.88	.69	.82	.60	**.84**	.39	.79	**.69**	.68	.73

Tablo 1'de görüldüğü gibi metotlar arasında çok büyük farklılıkları yoktur. Her metot farklı veri kümelerinde en iyi sonuçları elde etmiştir. Bu çalışmada en yüksek ortalama başarıya sahip olan *Centroid* metodu kullanılmıştır.

4. Deneysel Sonuçlar

Geliştirilen algoritma genel amaçlı olduğundan UCI [3] veritabanından alınan farklı özellikteki 15 adet veri kümesiyle test edilmiştir.

4.1. Farklı Data setlerin Üretilmesi

Bölüm 3'te anlatılan sebeplerle farklı data setlerin üretilmesinde rasgele alt uzaylar metodu kullanılmıştır. Orijinal özellik sayısına M denirse $\log_2 M$ ve $2\log_2 M$ olmak üzere iki farklı sayıda rasgele özellikler seçilerek veri alt kümeleri oluşturulmuş ve alt kümelerin her biri kümelendikten sonra kararları birleştirilmiştir.

Tablo 2'de kümeleme kararlarının birleştirilmesi denemelerinde kullanılan 15 UCI [UC] veri kümesi ve özellikleri verilmiştir.

Tablo 2: İki kümenin benzerliğini ölçen metotların karşılaştırılması

Dataset ID	Dataset İsmi	Özellik Sayısı	Sınıf Sayısı	Örnek Sayısı
1	Glass	9	6	170
2	derma	34	6	286
3	ecoli	7	8	266
4	breast-cancer	30	2	456
5	weather	34	2	281
6	iris	4	3	120
7	New-thyroid	5	3	143
8	segmentation	19	7	210
9	ionosphere	34	2	171
10	bupa	6	2	175
11	wine	13	3	118
12	waveform	21	3	2460
13	Monks1	6	2	124
14	Monks2	6	2	169
15	Monks3	6	2	122

4.2. Değerlendirme Kriteri

Elimizdeki veri kümelerindeki örneklerin sınıfları mevcut olduğundan kümeleme kararlarını birleştiren algoritmaların başarıları sınıflandırma performanslarıyla ölçülmüştür. Ayrıca rasgeleliği azaltmak için her algoritma her veri kümesine 10'ar kez uygulanmış ve sonuçların standart sapmaları da hesaplanmıştır.

Kümeleme algoritmalarının örneklerin sınıflarını vermediği bilinmektedir. Bunun için birleştirilmiş sonuçlardan yeni ortalamalar alınarak yeni küme merkezleri üretilmiş, merkezlere içerdiklerin örneklerin sınıflarına bakılarak sınıf etiketleri atanmıştır. Daha sonra merkezin etiketi içerdiği tüm örneklere aktarılmıştır. Bu sayede kümeleme sonuçlarının gerçek sonuçlarla aynı tür ve sayıda etiketlere sahip olması sağlanmıştır.

4.3. Sonuçlar

15 veri kümesinden elde edilen sonuçların ortalamaları, kararları birleştirilen kümeleyici türlerine, her bir kümeleyicinin küme sayısına göre ve özellik sayısına göre karşılaştırmalı olarak verilmiştir. Orijinal sütununda tüm veri kümesi tek bir kerede kümelendiğindeki başarı verilmiştir. Küme sayısı veri kümesindeki sınıf sayısına eşit ve 2 katı olacak şekilde 2 farklı şekilde seçilmiştir. Kümeleyicilerde 4 temel kümeleme algoritması kullanılmıştır. Kararların birleştirilmesinde her seferide 10'ar adet kümeleyici oluşturulmuş ve kararları iki farklı yöntemle (graf ve hiyerarşik) birleştirilmiştir. Tablo 3 ve Tablo 4'deki her değer (15 data set*10 tekrar sayısı) 150 değerin ortalaması alınarak elde edilmiştir.

Tablo 3'de küme sayısı sınıf sayısına eşit olduğu durumdaki sonuçlar görülmektedir. Tablo 3 incelendiğinde K-Means, Fuzzy K-Mean ve SOM algoritmalarının performansları arasında çok az bir fark vardır. Bununla birlikte ortalama olarak en başarılı algoritma Fuzzy K-Means'dir. Karar birleştirme sonuçları orijinal sonuçlardan neredeyse her zaman daha iyi sonuçlar üretmiştir. Bu da kümeleyici sonuçlarını birleştirmenin sınıflandırıcı sonuçlarını birleştirmedeki gibi performansa olumlu bir katkı yaptığını göstermektedir. Graf algoritmaları, hiyerarşik algoritmalardan daha yüksek bir ortalama başarıya sahiptir. Ancak bununla birlikte 15 veri kümesinde elde edilen en yüksek ortalama başarı **0,696** ile hiyerarşik algoritma sahiptir. Kümeleyicilerde kullanılan özellik sayısının artışının performansa az miktarda da olsa olumlu yansıdığı görülmektedir.

Tablo 3: Küme sayısı sınıf sayısına eşit olduğu durumdaki sonuçlar

cluster=sınıf	Orijinal	Hiye	graf	hiye	graf	
Özellik sayısı	M	$\log_2 M$	$\log_2 M$	$2\log_2 M$	$2\log_2 M$	Ort.
Kmeans	.673	.683	**.685**	.683	.684	.682
fuzzykmeans	.684	.692	.68	.69	**.692**	.688
SOM	.642	.683	.686	**.696**	.695	.68
eklemeli	.647	.624	.676	.64	.687	.654
Ortalama	.661	.67	.683	.677	.689	

Tablo 4'de küme sayısı veri kümelerindeki sınıf sayısının 2 katı olarak seçildiğindeki alınan sonuçlar verilmektedir.

Tablo 4: Küme sayısı sınıf sayısının 2 katı olduğu durumdaki sonuçlar

cluster=2*sınıf	Original	Hiye	graf	hiye	graf	
Özellik sayısı	M	Log2M	Log2M	2Log2 M	2Log2 M	Ort.
Kmeans	.75	.754	.74	**.755**	.742	.748
fuzzykmeans	.75	.749	.734	.753	.745	.746
SOM	.74	.722	.74	.737	.743	.736
eklemeli	.7	.683	.722	.697	.724	.705
ortalama	.735	.727	.734	.735	.738	

K-Means ve Fuzzy K-Means türünde kümeleyicilerin performans artışı sadece sonuçlarının hiyerarşi metoduyla birleştirildiğinde olmuştur. SOM'da ve hiyerarşikte ise performansı sadece graf metodu arttırmıştır.

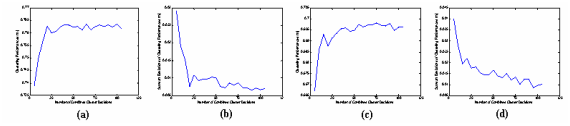
Tablo 5'de yapılan deneylerden çıkarılan sonuçlar karşılaştırmalı olarak verilmiştir.

Tablo 5: Özet Sonuçlar

	Sınıf sayısı= küme sayısı	2*Sınıf sayısı= küme sayısı
En başarılı kümeleme alg.	fuzzykmeans	Kmeans
En başarısız kümeleme alg.	eklemeli	eklemeli
En başarılı sonucu üreten karar birleştirme alg.	hiye-2log2M (SOM ile)	hiye-2log2M (Kmeans ile)
Ortalama en başarılı karar birleştirme alg.	graf-2log2	graf-2log2
Orijinal/ birleştirme sonuçlarından hangisi iyi:	orijinal< hiye<graf	hiye <orijinal<graf
Kümeleyicilerde kullanılan alt uzaylardaki özellik sayısının (boyutun) etkisi	Log2M< 2log2M	Log2M< 2log2M

Sınıf sayısının etkisi: Bölüm 4.3'de anlatılan kümeleme sonuçlarının sınıf sonuçlarına dönüştürülmesinde kullanılan metot dolayısıyla küme sayısının artışının başarıyı arttırması gayet doğaldır.

Kararları birleştirilen kümeleyici sayısının etkisinin de incelenmesi için 14 veri kümesi üzerinde denemeler yapılmıştır. Şekil 2.a ve c'de kümeleyicilerin performansları, b ve .d'de standart sapmaları verilmiştir. Verilen değerler 14 dataset üzerinde elde edilen değerlerin ortalamalarıdır. Şekil 2.a ve b'de her kümeleyicinin kullandığı özellik sayısı $2*\log_2 M$; küme sayısı sınıf sayısının iki katıdır. Şekil 2.c ve .d'te özellik sayısı $\log_2 M$, küme sayısı sınıf sayıyla aynıdır. Denemelerde kullanılan kümeleme algoritması Fuzzy K-Means'dir, Kümeleyicilerin kararlarının birleştirilmesinde hiyerarşik metot kullanılmıştır.



Şekil 2. Kümeleyici sayısının performansa ve standart sapmaya etkisi

Her iki tür denemede de kararları birleştirilen kümeleyici sayısı arttıkça performans artmakta, standart sapma azalmaktadır.

Kümeleyicilerde kullanılan özellik sayısının etkisinin daha iyi incelenebilmesi için ise aynı 14 veri kümesiyle denemeler yapılmış ve özellik sayısının $\log_2 M$'in 1 katından 5 katına kadar olan değerleri için performanslar ve standart sapmaları bulunmuştur. Her data set için yine 10'ar kez kümeleme yapılmış ve ortalamaları alınmıştır. Sonuç olarak özellik sayısı arttıkça genel olarak performansın arttığı, standart sapmanın azaldığı görülmüştür.

5. Tartışma

Sınıflandırma kararlarının birleştirilmesinin performansa olumlu katkılar yaptığını gören araştırmacılar aynı mantığım kümeleme algoritmaları içinde geçerli olabileceğini düşünmüşlerdir. Ancak kümeleme algoritmalarının sonuçlarının birleştirilmesi, sınıflandırmadaki kadar kolay değildir. Çünkü kümeleyici sonuçlarında küme sayısı sınıf sayısıyla aynı olmak zorunda değildir. Literatürde bu konuda çeşitli yöntemler önerilmiştir. Temel graf bölümleme ve hiyerarşik kümeleme olmak üzere iki tür yaklaşım mevcuttur. Genelde graf algoritmalarının daha başarılı sonuçlar ürettiği söylenmektedir.

Bu çalışmada bu iki türe ait metot 15 veri kümesi üzerinde 4 farklı kümeleme algoritması kullanılarak test edilmiştir. Kümeleme algoritmalarının, karar birleştirme algoritmalarının, kümeleme algoritmalarına verilen aynı veri kümesinden farklı alt kümeler oluştururken seçilen özellik sayısının etkileri incelenmiştir. Vardığımız sonuçlar aşağıdaki şekilde özetlenebilir:

✓Kümeleme yaparken, karar birleştirme yapmak genelde daha iyi sonuçlar üretmektedir. Ancak bu artış sınıflandırıcı algoritmalarındaki kadar belirgin değildir.

✓Ortalama olarak graf algoritmaları daha yüksek başarıya sahiptir ancak bireysel başarılarda hiyerarşik algoritmalar daha başarılıdır.

✓K-Means ve Fuzzy K-Means en başarılı kümeleme algoritmalarıdır.

✓Karar birleştirmenin başarısını en fazla arttırdığı kümeleme metodu eklemeli kümelemedir. Bunun sebebi eklemeli kümelemenin tekil başarısının diğerlerine göre daha düşük olmasıdır. Ancak eklemeli kümelemenin kararları birleştirildiğinde bile diğer algoritmaların performansına ulaşamamaktadır.

✓Kümeleyici komitelerinde yer alan kümeleyici sayısının artığı performansı arttırmakta ve standart sapmasını düşürmektedir.

✓Kümeleyicilerde kullanılan özellik sayısının arttırma performansı arttırmaktadır.

6. Kaynakça

[1] Jim C. Bezdek, "Fuzzy Mathematics in Pattern Classification" PhD Thesis, Applied Math. Center, Cornell University, Ithaca, 1973.

[2] Pang-Nıng Tan, Michael Steinbach, Vipin Kumar, "Introduction to Data Mining", Addison Wesley, 2005

[3] Blake, C., ve Merz, C.J., "UCI Repository of machine learning databases", 1998

[4] http://www.mathworks.com

[5] Kohonen,T., "The self-organizing map". Proceedings of the IEEE, 78(9), pp.1464–1480, 1990.

[6] Derek Greene, Alexey Tsymbal, Nadia Bolshakova, Padraig Cunningham, "Ensemble Clustering in Medical Diagnostics", 17th IEEE Symposium on Computer-Based Medical Systems, 2004

[7] X.Z. Fern, C.E. Brodley, "Solving cluster ensemble problems by bipartite graph partitioning", ACM International Conference Proceedings; Vol. 69(36), 2004.

[8] Alexander Strehl, Joydeep Ghosh, "Cluster ensembles - a knowledge reuse framework for combining multiple partitions", The Journal of Machine Learning Research, Vol. 3: 583-617, 2003.