ECOC Matris Optimizasyonu
Optimization of the ECOC Matrix

Cemre Zor

Mühendislik ve Fiziksel Bilimler Fakültesi
University of Surrey
c.zor@surrey.ac.uk

Berrin Yanıkoğlu

Mühendislik ve Doğa Bilimleri Fakültesi
Sabancı Üniversitesi
berrin@sabanciuniv.edu

ÖZETÇE

Hata Düzelten Çıktı Kodlaması (HDÇK) çok sınıflı sınıflandırma problemleri için bir sınıflandırıcı birleştirme yöntemidir. Bu yöntemde pekçok taban sınıflayıcı, önceden belirlenmiş bir kod matrisine göre, orijinal sınıfların farklı bir ikiye ayırma problemini öğrenir. Yeni bir veri geldiğinde elde edilen sınıflandırıcı çıktıları, her sınıf için kod matrisi tarafından belirlenmiş olan kod kelimesi ile, hata düzeltme kodlama yöntemi kullanılarak karşılaştırılır. HDÇK çok sınıflı sınıflandırma problemleri için en iyi yöntemlerden olsa da, bulunan çözüm optimal değildir çünkü kod matrisi ve taban sınıflandırıcılar birbirlerinden bağımsız belirlenir. Bu makalede bu ayrımı azaltıcı, yinelemeli bir algoritma önerilmektedir. Herbir yineleme, kod matrisinin eğitilen taban sınıflandırıcılar ile arasındaki farkı azaltıcı güncellemesinden oluşur. Yeni algoritmanın düz HDÇK'ya olan üstünlüğü bilinen birkaç veri tabanında gösterilmiştir.

ABSTRACT

Error Correcting Output Coding (ECOC) is a classifier combination technique for multiclass classification problems. In this approach, several base classifiers are trained to learn different dichotomies of the classes, specified by the columns of a code matrix. These classifiers' output for an unknown pattern is compared to the codeword of each class which is the desired output of the dichotomizers, in an error correcting fashion. While ECOC is one of the best solutions to multiclass problems, the solution is suboptimal due to the fact that the code matrix and the dichotomizers are set or learned independently. In this paper, we show an iterative update algorithm for the code matrix that is designed to reduce this decoupling. It consists of updates to the initial code matrix so as to reduce the discrepancy between the code matrix and the output of the trained dichotomizers. We show that the proposed algorithm improves over the basic ECOC approach, for some well-known data sets.

1. GİRİŞ

Hata Düzelten Çıktı Kodlaması (HDÇK) çok sınıflı sınıflandırma problemleri için bir sınıflandırıcı birleştirme yöntemidir [1]. Sınıflandırıcı birleştirme yöntemleri geçtiğimiz 10 senede çokça araştırılmış ve tek sınıflandırıcılara karşı avantajları hem teorik hem pratik açıdan ispatlanmıştır [2].

Tablo 1 Beş sınıflı bir problem için örnek kod matrisi.

	h1	h2	h3	h4	h5	h6
c1	+1	+1	+1	-1	-1	-1
c2	+1	-1	-1	+1	-1	-1
c3	+1	+1	-1	-1	-1	-1
c4	-1	-1	-1	+1	+1	-1
c5	-1	+1	+1	-1	+1	+1

HDÇK da çok sınıflı sınıflandırma problemleri için en iyi yöntemlerden biridir.

HDÇK yönteminde önce bir **kod matrisi** belirlenir. Bu matrisin her bir kolonu, bir taban sınıflayıcı için istenen sınıflararası bir ikiye ayırma problemini (dichotomy) tanımlar. Yani her sınıflandırıcı, sınıfların belli bir alt kümesini (örn. sınıfların ilk yarısını son yarısından ayırmak gibi) diğer sınıflardan ayırmaya çalışır. Yeni bir veri geldiğinde bütün sınıflandırıcılardan geçirilir ve elde edilen sınıflandırıcı çıktısına en yakın **kod kelimesine** sahip olan sınıfa atanır.

Bu işlemin detayları şöyledir. Elimizde bir tane $K \times L$ C isimli bir kod matrisimiz olduğunu varsayalım. Burada K sınıf sayısı ve L sınıflandırıcı sayısıdır. Kod matrisin herhangi bir C_{ij} elemanı, i sınıfı için, j'inci sınıflandırıcının eğitilmesi sırasında istenen (+1,-1) çıktısını belirler. Bu şekilde belirlenen kod matrisinin herbir satırı o sınıfın kod kelimesini oluşturur. Benzer şekilde, kod matrisinin her kolonu da o sınıflandırıcı için seçilmiş olan sınıf ikiye ayırma problemini belirtir. Tablo 1 5-sınıflı ve 6 sınıflandırıcılı bir problem için tasarlanan bir kod matrisini gösterir. Bu kod matrisi, h1 sınıflandırıcısının C1-C3 sınıflarını C4-C5 sınıflarından, h6 sınıflandırıcısının ise C5'i diğer tüm sınıflardan ayıracak şekilde eğitilmesini belirtir.

Sınıflandırıcı çıktısı ve kod kelimeleri arasındaki uzaklık Hamming veya Öklid uzaklığı ile ölçülür. En yakın kod kelimesinin seçilmesi ile taban sınıflandırıcıların bir miktar hatası düzeltilebilir. Tam olarak, sınıf çiftleri arasındaki Hamming uzaklığına d dersek, test çıktısındaki $\lfloor (d-1)/2 \rfloor$'ye kadar bit hatası düzeltilebilir.

Kod matrisleri farklı şekilde belirlenebilir; örneğin yaygınca kullanılanlar arasında rastgele, tek sınıfı diğerlerinden ayıran (one against all), ve çiftleri birbirinden ayıran kod (pairwise) matrisler kullanılabilir. Kod matrisi belirlemede genel olarak dikkat edilmesi gereken nokta, çözümleme sırasındaki hata düzeltmeye yardım etmesi için sınıflar arasında büyük Hamming uzaklığı (HU) olmasıdır. Aynı zamanda, sınıflandırıcı birleştirme sistemlerinde gerektiği üzere,

kolonlar arası HU'nun da yüksek olması gerekir ki taban sınıflandırıcıların birbiri ile korelasyonu az olsun. Bu sebeple bootstrapping de sınıflayıcıların eğitileceği veri kümelerini farklı kılmak için yaygınca kullanılır. Kod matrisi nasıl belirlenmelidir diye çok çalışma olsa da [3,4], sınıflandırıcılar eğitildikten sonra kod matrisini iyileştirme yönünde çok az çalışma vardır [5].

Ana probleme göre daha basitleştirilmiş olsa da, taban sınıflandırıcıların kod matrisi ile belirtilen görevi yine de genellikle çok basit değildir. Hata düzeltme yöntemi kullanılsa da, kod matrisin nasıl belirlenmesi üstünde çalışılan bir konudur. Örneğin 0 hedefinin de bulunduğu sıfırlı kod matris dizaynı, 0 verilen sınıfları sınıflandırma dışında bırakarak taban sınıflandırıcısının işini kolaylaştırmaya çalışır [6,7].

Kod matrisinin nasıl belirlenmesine ek olarak daha genel bir sorun, HDÇK'larda kod matrisi ve taban sınıflandırıcıların birbirlerinden bağımsız belirlenmesi veya eğitilmesidir. Yani HDÇK ile sınıflandırıcı birleştirimi bütün olarak optimal değildir. Bu makalede bu ayrımı azaltıcı yinelemeli bir algoritma önerilmektedir. Herbir yineleme, kod matrisinin eğitilen taban sınıflandırıcılar ile arasındaki farkı azaltıcı güncellemesinden oluşur. Bu yöndeki bir çalışmamızda, kod matrisin eğitilmiş olan sınıflandırıcılara göre eniyilenmesinin başarıyı artırdığını göstermiştik [10]. Bu makalede de daha önce önerilen bu algoritmanın, üçlü kod matrisleri kullanarak daha da genişletilmesi sağlanmıştır. Yeni algoritmanın düz HDÇK'ya olan üstünlüğü bilinen birkaç veri tabanında gösterilmiştir. Makalenin devamında HDÇK yerine yöntemin daha bilinen kısaltması olarak ECOC kullanılmaktadır.

2. ÖNERİLEN YÖNTEM

Elimizde bir C kod matrisi ve buna göre eğitilmiş taban sınıflandırıcıları olsun. Burada C_{ij} j sınıflandırıcısının i sınıfı için hedefidir. Ek olarak A_{ij}'de buna karşılık gelen başarı değeri olsun (j sınıflandırıcısının i sınıfındaki başarısı). Bu başarıyı geçerleme (validation) kümesindeki c_i örneklerinin j sınıflayıcısı tarafından, doğru C_{ij}'de belirtilen hedef değere göre doğru tanınma oranı olsun. Önerilen yöntem şu temel gözleme dayanır. Diyelim ki bir taban sınıflandırıcısı h_j, +1 olarak tanıması gereken c_i sınıfında %0 başarılı. Kod matrisindeki ilgili hücre C_{ij}'yi +1'den -1'e değiştirmemiz sınıflandırıcıyı kod matrisine uyduramadığımız bir durumda, kod matrisini sınıflandırıcıya uydurmaya karşılık gelir. Yani h_j diğer bütün sınıflardaki başarısı aynı kalacak şekilde, c_i sınıfı için %100 başarıya ulaşır. Tabi normalde böyle uç başarı değerleri olmaz, ama bu durum önerilen metodu anlatmak için seçilmiştir.

Peki bu değişimin etkisi genel sınıflamaya nasıl yansır? Bunun için şunu görmek lazım ki, bu sınıflandırıcı için, başarısız olunan sınıf ile farklı hedef değerine (-1) sahip olan sınıflar ve aynı hedef değerine (+1) sahip sınıflar vardır. Kod matrisinde yapılacak değişiklik c_i'nin birinci grupla olan Hamming Uzaklığını azaltır, ancak bunun genel sınıflandırmaya katkısı kötü olmaz çünkü bir test girdisi için elde edilen çıktıya o dersek, $o[i]$ bit'i her zaman yanlış olacak ve c_i sınıfına uzaklığı artıracaktır. C_{ij} değişince sadece bu durum ortadan kalkar; bu bit'ten diğer sınıflara olan uzaklık katkısı ise aynı kalır çünkü o ve C_{kj} değişmez.

Algoritma 1 Önerilen ECOC Optimizasyonu

A ve C matrislerini hesapla
C_{ij}'yı $A_{ij} < 0.5$ artan sırada olacak şekilde listele;
elemanSayisi ← listedeki eleman sayısı;
$\beta = 0.3$
$\alpha = 0.6$
for $t = 1$: elemanSayisi **do**
  {t'inci en kötü sonuca karşılık gelen C[t]'yi değiştir}
  $C \leftarrow$ Şu anki ECOC matrisi;
  $yeniC \leftarrow$ C;
  if $A_{ij} < \beta$ **then**
    $yeniC[t]$'yi ters çevir; {FLIP}
  else if $A_{ij} < \alpha$ **then**
    $yeniC[t]$'yi sıfırla; {ZERO}
  end if
  {Geçerleme/Validasyon başarısı artarsa, değişikliği kabul et}
  $\Delta kazanç \leftarrow$valAccuracy[$yeniC$]−valAccuracy[C];
  if $\Delta kazanç \geq 0$ **then**
    $C \leftarrow yeniC$;
  end if
end for

Önerilen yöntemde, biz belli bir başarı değerinin altındaki A_{ij} değerlerine karşılık gelen C_{ij} değerlerini tersine çevirmeyi (flip) veya sıfırlamayı (zero) öneriyoruz. Sezgisel olarak da mantıklı gelecek bir sonuç, çok düşük A_{ij} değerlerinde tersine çevirme, daha yüksek değerlerde sıfırlamanın genellikle daha iyi sonuç vermekte olduğudur. Bu güncelleme işlemi, en kötü performansı veren A_{ij} değerlerinden en iyiye doğru yapılmaktadır. Bu çalışma, daha önce geliştirdiğimiz FLIP-ECOC algoritmasının [10], önerilen sıfırlama durumu ile, 3'lü kodlara genişletilmiş halidir.

Dikkat edilmelidir ki tersine çevirme veya sıfırlama tümüyle sorunsuz değildir, çünkü satırlar ve sütunlar arası HU'yu kolayca bozabilir. Bu da tek bir sınıfta kazanılacak görece küçük başarı artışından çok daha kötü olabilir. Bu sebeple en düşükten en yükseğe doğru sıralanan A_{ij} değerlerine bakıldığında, en düşükten başlayarak tersine çevirme veya sıfırlama işlemi yapılır; sonuç geçerleme kümesinde denenir ve genelde iyi ise kabul edilir. Bu açgözlü (greedy) algoritma Algoritma 1'de sözde-kod ile verilmiştir.

FLIP-ECOC'a göre bu makalede yapılan katkı, sıfırlama ile üçlü kod matrisi yapısına geçilmesidir. ECOC matrisi belirlerken 0 hedef, bu sınıfa bakma (don't care) demektir ve birbirinden kolaylıkla ayrılamayacak sınıflarda çok etkilidir. Üçüncü bir hedefin, yani sıfırın kulanılmasının önerdiğimiz algoritmada ki katkısı da bu yöndedir. Yani zaten üçlü kod matrisi kullanan sistemlerde, üçlü kod matrisinin getirdiği avantajlar önerilen algoritmada da görülür. Yani sınıflandırıcılar eğitildikten sonra, eğer başarı oranı, sınıfın herhangi bir hedefe (+1 veya -1) atanmasını savunamayacak gibi ise, örneğin 0.5 ise, o zaman sıfırlama çok işe yarar çünkü o bit'in HU sırasında dikkate alınmamasını sağlar.

Ancak sıfırlı kod matrislerde kod çözme (decoding) çok basit değildir. Çünkü test çıktısı o ile bir sınıfın kod kelimesi arasındaki HU hesaplanırken, bu uzaklık daha az sıfırlı bir

kod kelimesinden gelmişse daha bilgilendiricidir. Dolayısıyla aynı HU değeri veren 2 kod kelimesinden hangisinin, nasıl tercih edileceğine karar verilmesi gerekebilir. Sıfırlı kod matrisler için farklı çözümleme yöntemleri araştırma konusudur [7,8]. Bu çalışmada HU basitçe sıfır olmayan bitler üzerinden hesaplanmıştır, ancak daha akıllıca yöntemler denenebilir.

3. Sonuçlar

Önerilen yöntemin temel ECOC'a ve FLIP-ECOC'a üstünlüğü önceki deneylerimizde kullandıklarımız arasında rastgele seçilen ve Tablo 2'de verilen 3 UCI veri kümesinde [9] denenmiştir. Ayrı test kümesi olan veri setlerinde (Satellite), eğitim kümesi eşit 2 parçaya rastgele ayrılmıştır (eğitim ve geçerleme kümeleri) ve sonuçlar 10 farklı yürütme (run) ortalaması şeklinde elde edilmiştir. Test kümesi olmayan veri setlerinde ise 10 kereli çapraz sağlamanın ortalaması şeklinde elde edilmiştir.

Bu deneylerde ECOC kod matrisi olarak 10, 25, 75 veya 150 kolonluk (K) (taban sınıflandırıcı sayısı) matrisler kullanılmış; sınıf sayısı veri kümesinden belirlenmiş; ve taban sınıflandırıcılar olarak 2, 8, veya 16 düğümlü (D) Yapay Sinir Ağları kullanılmıştır. ECOC yaklaşımında onlarca veya yüzlerce sınıflandırıcının eğitilmesi oldukça zaman alan bir iş olduğundan, bu konuda çabuk eğitilebilecek zayıf (weak) sınıflandırıcıların katkısını denemek istedik. Bu amaçla, taban sınıflandırıcıların 15 tur (epoch) eğitilmeleri yanında, sadece 2 tur eğitilmelerini de parametre olarak değerlendirdik. Dolayısıyla tablolardaki ikinci sütundaki (10K-2D-15E) sonuçlar, 10 kolonlu, 2 düğümlü, ve 15 tur boyunca eğitilmiş 10 yürütmenin ortalamasıdır.

Deneysel sonuçların özeti Tablo 3-5'te verilmiştir (benzer sonuçlar veren 8 düğümlü ve 25 kolonlu sonuçlar tablolara dahil edilmemiştir). Bu sonuçların hemen hepsinde önerilen sistem ve normal ECOC arasında belirgin; önerilen sistem ve FLIP-ECOC arasında da küçük başarı artışı görülmektedir. FLIP-ECOC'a göre elde edilen ortalama sonuçlar, 18 farklı parametre değerinin 10'unda artış, 2'sinde kötüleşme, 6'sında aynı kalma şeklinde özetlenebilir. Artış ve azalmalar Tablo 3-5'te sırasıyla koyu veya eğik olarak belirtilmiştir. Önerilen yöntem özellikle zayıf sınıflayıcılarla gayet başarılı olmaktadır, ancak sınıflayıcıların ortalama başarısı düştükçe iyileştirilebilecek alan ufaldığı için önerilen yöntemin katkısı sıfıra kadar azalmaktadır.

Önerilen ECOC matris optimizasyon yöntemleri (FLIP-ECOC ve bu makalede anlatılan geliştirilmiş hali), başarıyı artırmalarının yanında, fazladan eğitim gerektirmedikleri için her tür ECOC tabanlı sınıflandırıcı birleştirme sisteminde denenebilir. Gereken sadece bir geçerleme kümesinde yapılan bit değişikliklerinin işe yarayıp yaramadığına bakılmasıdır.

4. Yapılacak İşler

Üçlü kod matrisinin başarısı çözümlemede sıfır bitlerinin nasıl ele alınacağına dayanır. Biz bu çalışmada basit şekilde HU kullandık, halbuki daha farklı çözümleme yöntemleri, sıfırlı ECOC yöntemini iyileştirdiği gibi, öneilen algoritmanın FLIP-ECOC'a farkını da artırabilir.

Şu anda sistem bitleri en kötü performans verenden başlayarak değiştirmektedir, ancak bu durumda değişiklikler kod matrisinin bir sırasında (kötü tanınan bir sınıf) veya bir kolonunda (kötü performanslı bir taban sınıflayıcı) toplandıklarından, sınıflar veya sınıflandırıcılar arası Hamming uzaklığını çok azalttıklarından dolayı başarı artışı olabileceği kadar yüksek olmayabilir. Bunun için bu uzaklıkları da gözönüne alan bir yenileme algoritması düşünülebilir.

Ortalamada denenen veri kümelerinde FLIP-ECOC'a göre ufak iyileşmeler gözlenmektedir; ancak sistemde kullanılan α ve β parametrelerinin, belki de sınıflandırıcının ortak başarısına göre belirlenmesi daha başarılı sonuçlar getirebilir.

Sistemin ayrıca farklı veri kümeleri ile değerlendirilmesi ve sonuçların istatistiksel olarak belirgin olup olmadığına bakılmalıdır.

5. KAYNAKÇA

[1] Dietterich, T.G., Bakiri, G.: Solving Multi-class Learning Problems via Error- Correcting Output Codes. J. Artificial Intelligence Research 2. 263–286 (1995).

[2] James, G. M.: Majority Vote Classiffiers: Theory and Applications. PhD Thesis, Department of Statistics, University of Standford (1998)

[3] Escalera, S., Tax, D. M. J., Pujol, O., Radeva, P., Duin, R. P. W.: Subclass Problem-Dependent Design for Error-Correcting Output Codes. In: IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 30, no. 6, pp. 1041-1054 (2008)

[4] Escalera, S., Pujol, O., Radeva, P.: Recoding Error-Correcting Output Codes. Proceedings of the 8th International Workshop on MCS, vol. 5519, pp. 11–21 (2009)

[5] Alpaydin, E., Mayoraz, E.: Learning error-correcting output codes from data. In: Proc. Int. Conf. Neural Networks (ICANN) (1999).

[6] Allwein, E., Schapire, R., Singer, Y.: Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers. JMLR 1. 113–141 (2002).

[7] Escalera, S., Pujol, O., Radeva, P.: On the Decoding Process in Ternary Error- Correcting Output Codes. In: CIARP, vol. 4225, pp. 753–763 (2006)

[8] Windeatt, T., Ghaderi R.: Coding and Decoding Strategies for Multi-class Learning Problems. Information Fusion, 4(1), pp. 11-21 (2003)

[9] Asuncion, A., Newman, D.J.: UCI Machine Learning Repository, http://www.ics.uci.edu/ mlearn/MLRepository.html. School of Information and Computer Science, University of California, Irvine, CA (2007).

[10] Zor, C., Yanikoglu, B., Windeatt, T., Alpaydin, E.: FLIP-ECOC: a greedy optimization of the ECOC matrix. ISCIS 2010, (2010).

Tablo 2 Deneylerde kullanılan 3 UCI MLR veri kümesinin özeti.

	Eğitim Kümesi Büyüklüğü	Test Kümesi Büyüklüğü	Özellik Sayısı	Sınıf Sayısı
Glass Identification	214	-	10	6
Dermatology	358	-	33	6
Satellite Image	210	2100	19	6

Tablo 3 Dermatology Veri Kümesi Sonuçları.

	10K-2D-2E	10K-2D-15E	75K-2D-2E	75K-2D-15E	150K-2D-2E	150K-2D-15E	Ortalama
Normal ECOC	74,02	93,13	96,15	97,24	96,40	97,40	92,85
FLIP-ECOC[10]	83,40	93,47	97,07	97,07	97,07	97,74	94,87
ÖNERİLEN	**84,66**	**94,14**	97,07	97,07	97,07	97,74	**95,27**

Tablo 4 Glass Veri Kümesi Sonuçları

	10K-2D-2E	10K-2D-15E	75K-2D-2E	75K-2D-15E	150K-2D-2E	150K-2D-15E	Ortalama
Normal ECOC	38,70	55,64	51,89	64,15	55,92	63,74	55,53
FLIP-ECOC	48,91	56,91	59,13	65,27	60,09	65,43	59,59
ÖNERİLEN	48,92	56,91	*58,12*	**65,82**	**60,80**	**66,66**	**59,64**

Tablo 5 Satellite Veri Kümesi Sonuçları

	10K-2D-2E	10K-2D-15E	75K-2D-2E	75K-2D-15E	150K-2D-2E	150K-2D-15E	Ortalama
Normal ECOC	56,18	79,89	73,99	84,76	79,37	85,14	78,62
FLIP-ECOC[10]	70,68	80,76	82,46	84,88	82,84	85,24	82,19
ÖNERİLEN	*67,79*	**81,17**	**83,05**	**84,93**	**83,41**	**85,40**	**82,39**

UCI Dermatology Veri Kümesi Sonuçları[1]

	10K-2D-2E	10K-2D-15E	75K-2D-2E	75K-2D-15E	150K-2D-2E	150K-2D-15E	Ortalama
Normal ECOC	74,02	93,13	96,15	97,24	96,40	97,40	92,85
FLIP-ECOC	83,40	93,47	97,07	97,07	97,07	97,74	94,87
FLIP-ECOC$^+$	**84,66**	**94,14**	97,07	97,07	97,07	97,74	**95,27**

UCI Glass Veri Kümesi Sonuçları

	10K-2D-2E	10K-2D-15E	75K-2D-2E	75K-2D-15E	150K-2D-2E	150K-2D-15E	Ortalama
Normal ECOC	38,70	55,64	51,89	64,15	55,92	63,74	55,53
FLIP-ECOC	48,91	56,91	59,13	65,27	60,09	65,43	59,59
FLIP-ECOC$^+$	48,92	56,91	*58,12*	**65,82**	**60,80**	**66,66**	**59,64**

UCI Satellite Veri Kümesi Sonuçları

	10K-2D-2E	10K-2D-15E	75K-2D-2E	75K-2D-15E	150K-2D-2E	150K-2D-15E	Ortalama
Normal ECOC	56,18	79,89	73,99	84,76	79,37	85,14	78,62
FLIP-ECOC	70,68	80,76	82,46	84,88	82,84	85,24	82,19
FLIP-ECOC$^+$	*67,79*	**81,17**	**83,05**	**84,93**	**83,41**	**85,40**	**82,39**