

İSTENMEYEN İLETİLERİN PARALELLEŞTİRİLMİŞ KNN ALGORİTMASI İLE TESPİTİ

Tuğba YILDIZ*, Savaş YILDIRIM*, Yrd. Doç. Dr. D. Turgay ALTILAR**

(*) İstanbul Bilgi Üniversitesi, Bilgisayar Bilimleri Bölümü, İSTANBUL

(**) İstanbul Teknik Üniversitesi, Bilgisayar Mühendisliği Bölümü, İSTANBUL
tdalyan@cs.bilgi.edu.tr, savasy@cs.bilgi.edu.tr, altilar@itu.edu.tr

ÖZET

Elektronik posta kullanımının arttığı günümüzde, istenmeyen iletilerin sayısında artmıştır. Çalışma, elektronik iletilerin belirlenmesi için kişisel bir filtreleme modeli geliştirilmesine yöneliktir. Bunun için geliştirilen sınıflandırıcı sayesinde, gelen iletinin istenen ya da istenmeyen ileti olup olmadığına karar verilir. Bu kararın verilme süresinin kısa olması gerektiği düşünülürse, sınıflandırıcının sonucu bildirme süresinin kısa olması gerekmektedir. Çalışmada, sürenin kısılması için, geliştirilen sınıflandırıcının paralelleştirilmesi sağlanmıştır.

Anahtar Kelimeler: k-en yakın komşu algoritması, paralelleştirme, mpiJava.

SPAM FILTERING WITH PARALELLIZED KNN ALGORITHM

ABSTRACT

Usage of electronic mail is increasing day by day and at the same number of spam mail is also increasing. This study intended for developing a individual filtering model to determine whether the e-mail is spam or not. Classifier, which is implemented, is useful to make decision. Because of long time, parallelism of classifier is provided.

Keywords: k-nearest neighborhood, parallelizing ,mpiJava.

1. GİRİŞ

Günümüzde, elektronik posta (e-posta) ile iletişim kurma hayatımızın vazgeçilmezi haline gelmiştir. E-posta kullanımı arttıkça, kullanıcı tarafından istenmeyen ve farklı amaçlar içeren elektronik iletilerin sayısında artmıştır. Bu tip maillerden korunmaya yönelik programlar geliştirilse de ya da birçok e-posta programları istenmeyen iletiler ile başa çıkmak için araçlar geliştirse de çoğu yetersiz kalmaktadır.

İstenmeyen iletilerin belirlenmesine yönelik birçok veri madenciliği çalışması da yapılmıştır [1]. Karar ağaçları [2], kural çıkarımı [3], bayesian sınıflandırıcılar [4,5] yanında, genetik algoritmalar [6] ve destek vektör makinaları [7] gibi yaklaşımlarda kullanılmıştır. Bunun yanında, Data Mining Cup [8] gibi yarışmalara da konu olmuştur.

Bu çalışmada da, 2006 yılında, ECML/PKDD tarafından [9], istenmeyen elektronik iletilerin belirlenmesine yönelik olan yarışmadaki amaç ile yola çıkılmıştır. Yarışmada, A görevi adı altında sunulan veriler kullanılarak kişisel bir filtreleme modeli tasarlanmaya çalışılmıştır.

Çalışmada, sınıflandırıcı olarak KNN algoritması seçilmiştir. Yarışma [9] tarafından belirlenen veri kümeleri üzerinde, sınıflandırıcı çalıştırılarak sonuçlar elde edilmiştir. Sonuçların kısa sürede elde edilmesi açısından, sınıflandırıcının paralelleştirilmesi sağlanmıştır.

Makalenin ilk kısmında, kullanılan veri kümeleri ve algoritmanın işleyişinden bahsedilmiştir. Üçüncü bölümde, sınıflandırıcının paralelleştirilmesine yönelik yapılan çalışmalar üzerinde durulmuştur. Dördüncü bölümde, deneysel sonuçlara yer

verilmiştir. Son kısımda ise sonuçlar yer almaktadır.

2. VERİ KÜMELERİ VE ALGORİTMANIN İŞLEYİŞİ

Bu bölümde kullanılan veri kümeleri ve algoritmanın işleyişi hakkında bilgi verilmiştir.

2.1. Kullanılan Veri Kümeleri

Yarısmada sunulan öğrenme kümesi, %50'si istenmeyen, %50'si istenen olmak üzere 4000 e-iletiden oluşur.

Dosyanın yapısı şu şekildedir:
1 9:3 94:1 109:1 163:1 405:1 406:1 415:2
-1 2:3 9:8 17:3 35:7 71:3 74:2 77:6 85 ...

İlk sütun, verinin istenen ya da istenmeyen ileti olduğunu, iki nokta ile ayrılan verilerden, soldaki kelime numarasını, sağdaki ise, o kelimenin kaç defa görüldüğü bilgisini içermektedir.

Sınama kümesi ise 3 farklı kişinin, gelen kutularından oluşan etiketlenmemiş veri kümelerinden oluşur. Sınama verisinde ilk sütun etiketlenmemiş olduğundan 1 ya da -1 yerine, 0 sayısı içermektedir. Amaç, öğrenme verisini kullanarak oluşturulan modelden, 0 yerine gelebilecek değeri bulmaktır. Modelin doğruluğunu test etmek için, sınama kümelerinin doğru etiketlenmiş halleride ayrı verilmektedir. Bunlara ek olarak, model parametrelerini ayarlamak üzere kullanılacak bir adet etiketlenmiş geçereleme veri kümesi verilmiştir.

2.2. Algoritmanın İşleyişi

Çalışmada, sınıflandırıcı olarak K-en yakın komşu (KNN) algoritması kullanılmaktadır. KNN, eğitilmiş öğrenme algoritmasıdır ve amacı, yeni bir örnek geldiğinde varolan öğrenme verisi üzerinde sınıflandırma yapmaktır. Algoritma, yeni bir örnek geldiğinde, onun en yakın K komşusuna bakarak örneğin sınıfına karar verir. [10]

Örneğin, x ve y isimli iki niteliğimiz olduğunu ve Tablo 1 'deki gibi değerleri olduğunu varsayalım.

Tablo 1: X ve Y niteliğinden oluşan veri kümesi

X	Y	Sınıf
10	12	Evet
15	10	Hayır
5	10	Evet
14	12	Hayır

X niteliğine ait 12, Y niteliğine ait 10 değerine sahip yeni bir örnek geldiğini düşünürsek, şu adımları gerçekleştirmemiz gerekmektedir:

1. K değerini seç
2. Tüm öğrenme örnekleri ile olan uzaklığını hesapla
3. Minimum uzaklığa göre sırala
4. Ait oldukları sınıf değerlerini bul
5. Değeri baskın olan sınıfı seç

İlk adım olarak, K değerini 3 seçtiğimizi düşünelim. İkinci adımda uzaklık hesabı yapmamız gerekmektedir. Bunun için uzaklık ölçütlerinden biri kullanılabilir. En çok kullanılan ölçüt Euclid Uzaklık Ölçütüdür.

Üçüncü adımda küçükten büyüğe doğru sıralanarak, minimum k değer alınır. Tablo 2'nin dördüncü sütununda vurgulanmaktadır. Dördüncü adımda ise ait oldukları sınıf değerleri bulunur. Son adımda ise seçilen örneklerin maksimum sayıda olanı alınır. Bu durumda 2 hayır, 1 evet olduğuna göre, sonuç hayır olarak alınır belirlenir.

Tablo 2. KNN algoritmasındaki adımlar

X	Y	(12, 10) için olan uzaklık	Sıralama	Ait oldukları Sınıflar
10	12	$(10-12)^2 + (12-10)^2 = 8$	1	Evet
15	10	$(15-12)^2 + (10-10)^2 = 9$	2	Hayır
8	10	$(8-12)^2 + (10-10)^2 = 16$	4	-
15	12	$(15-12)^2 + (12-10)^2 = 13$	3	Hayır

Çalışmada da standart KNN algoritması kullanılmıştır. Yukarıdaki adımlar üzerinden gidersek:

1. K değerinin 3 olarak seçildiğini varsayalım. K değerinin seçilmesine yönelik yapılan çalışmalar ileriki bölümlerde anlatılacaktır.

2. Herbir sınaama örneği için, öğrenme kümesindeki tüm örnekler ile olan uzaklıklar, euclid uzaklık ölçütü ile hesaplanmıştır. Böylelikle, bir sınaama kayıdı için, öğrenme kümesindeki 4000 kayıt ile oluşturulan uzaklık listesi oluşturulmuştur. Örneğin:

Öğrenme kümesi için oluşturulan liste:

[[[-1,2,3,9,8,17,2],[1,9,3,94,1,109,1],

[1,2,2,3,1,9,10,12,2],...]

Sınaama kümesi için oluşturulan liste :

[[[0,2,4,9,13,10,1,11,1],...]

için oluşacak listenin elemanları [32,120,20...] şu şekildedir. Bu liste 4000 adet eleman içermektedir. Çünkü bir adet sınaama örneği için 4000 adet öğrenme örneği ile uzaklık hesaplanır. Formule uyularak, karelerinin farkı alınır fakat, karekök işlemi yapılmaz. Karelerinin farkı işimizi göreceği için ve karekök işlemi içinde bir süre kullanılacağı için, karekök işlemi gözardı edilir.

3. Oluşan bu liste küçükten büyüğe doğru sıralanır, sıralanırken sayıların indisleride de ayrı başka bir listede tutulur. Yukarıdaki örneğe göre:

Sıralı liste : [20,32,120,...]

İndis listesi : [2,0,1]

4. Oluşan sıralı dizinden belirlenen ilk k adet sayı belirlenir. k değerinin 3 olduğunu düşünürsek, işimize yarayan listeler indislerin bulunduğu listenin ilk 3 elemanıdır. İndisler sayesinde, istenen ya da istenmeyen ileti olup olmadığı bilgisini bir listeye atar. Oluşan listedeki istenmeyen ileti sayısı fazla ise, sınaama örneğine -1, istenen ileti sayısı fazla ise sınaama örneğine 1 değeri atanır. Eğer eşit sayıda ise değerler rastgele atanır.

Bu durumda ileti listesi : [1,-1,1]

5. Son adımda ise, listedeki 1 sayısı fazla olduğu için ilk sınaama örneğinin etiket değeri 1 olarak atanır.

Yukarıda da anlatıldığı gibi KNN algoritmasının tüm adımları gerçekleştirilmiştir. Bunun dışında, programda değerlendirme adımı da gerçekleştirilmiştir. Değerlendirme adımı, karışıklık matrisinden yararlanılarak, doğruluk, hata, anma, kesinlik ve f-ölçütü değerleri bulunur. Aşağıdaki Tablo 3. de karışıklık matrisi verilmiştir.

Tablo 3. Karışıklık Matrisi

DOĞRU SINIF	ÖNGÖRÜLEN SINIF		TP: a FN: b FP: c TP: d
	Sımf= 1	Sımf=-1	
Sımf=1	a	b	
Sımf=-1	c	d	

Buna göre;

$$\text{doğruluk} = ((TP+TN)/(TP+TN+FN+FP))$$

$$\text{hata} = ((FP+FN)/(TP+TN+FN+FP))$$

$$\text{kesinlik} = (TP/(TP+FP))$$

$$\text{anma} = (TP/(TP+FN))$$

$$\text{Fölçütü} = ((2 * \text{kesinlik} * \text{anma}) / (\text{kesinlik} + \text{anma}))$$

Değerlendirme kısmı programda şu şekildedir:

6. Yukarıdaki adımlar sonucunda 2500 sınaama örneği için, elimizde öngörülen 2500 etiket listesi bulunur. Yarışma, sınaama örneklerinin gerçek etiketlerini de sitesinde yayınladığı için, elimizde 2500 adet de doğru olarak etiketlenmiş etiket listesi vardır. Örneğin:

8 adet sınaama örneği için öngörülen etiket listesi : [1, -1, 1, 1, -1, -1, 1, 1]

8 adet sınaama örneği için doğrulanmış etiket listesi : [1, -1, 1, -1, 1, -1, 1, 1] ise

$$\text{doğruluk} = 6/8 = 0.75$$

$$\text{hata} = 2/8 = 0.25$$

$$\text{kesinlik} = 4/5 = 0.8$$

$$\text{anma} = 4/5 = 0.8$$

$$\text{F-ölçütü} = 0.8$$

Bu yöntem kullanılarak, değerlendirme işlemi yapılmıştır.

2.3 k Değerinin Belirlenmesi

Programda kullanılacak k değeri, yarışma tarafından sunulan geçerleme veri kümesi üzerinde denemeler yapılarak belirlenmiştir. k'nın 2'den 10'a kadar olan tüm değerleri denenerek en yüksek doğruluk oranının hangi değerde verildiği saptanmaya çalışılmıştır. Bu testler sonucunda, k değerleri arasında çok büyük fark görülmesi de, k'nın 2 ve 5 değerleri için doğruluk oranının diğerlerine göre daha yüksek olduğu görülmüştür.

2.4 Eşik Değerinin Belirlenmesi

Belirlenen bir eşik değeri ile işlem süresinin kısaltılması sağlanmıştır. Eşik değerinin belirlenmesi de, k değerinin belirlenmesi gibi geçerleme kümesi kullanılarak belirlenmiştir. Bölüm 2.3'de anlatıldığı gibi, geçerleme verisi üzerindeki deneylerde, k değeri 2'den 10'a kadar seçilmiş ve sıralama işleminden sonra k değerine göre yaklaşık bir eşik değeri belirlenmiştir. Bu da en yakın k için belirlenenin değerinin üstündeki değerler için önemsiz olduğunu göstermektedir.

3. SINIFLANDIRICININ PARALELLEŞTİRİLMESİ

Çalışmada, karşılaşılan en büyük sorunlardan biri veri kümesinin yapısından kaynaklanmıştır. Veri kümelerindeki her örnek için nitelikler, nitelik sayısı ya da nitelik değerleri farklılık göstermektedir. Bu da standart bir işlem yapmaktansa, hesaplama yapılırken, birçok koşulun oluşmasına ve tüm oluşan koşullarda, programın süresinin artmasına sebep olmuştur. Gelen bir e-postanın istenen ya da istenmeyen bir ileti olduğuna hemen karar verilmesi gerektiği düşünülürse, uzun bir işlem kullanıcı açısından kullanışlı olmamaktadır.

Çalışmada karşılaşılan diğer bir sorun ise algoritmanın yapısından kaynaklanmaktadır. Bilindiği gibi KNN algoritmasının en önemli avantajlarından biri

model yaratmamasıdır. Bu sebeple eğitim için bir zaman harcamaz. Ama bu durum aynı zamanda bir dezavantaj yaratmaktadır. Çünkü algoritma sınama için diğer modellerin aksine daha çok zamana ihtiyaç duyar. Örnek olarak 4000 boyutlu eğitim kümesi düşünelim. Eğer KNN algoritması uygulanırsa bir tek sınama örneği için 4000 satırın herbiriyle karşılaştırma yapılır. Bunun aksine karar ağaçları gibi model yaratan algoritmalar kullanırsa, en fazla söz konusu problemin özellik sayısı kadar bir işlem uygulanır. Bu sebepten KNN algoritması yeni gelen örnekleri olumlu olumsuz olarak belirlemesi için çok fazla zamana ihtiyaç duyar.

Bunları gidermek için paralelleştirme işlemi ile harcanan zamanı azaltmak hedeflenmiştir.

Algoritma java programlama dilinde gerçekleştirildiği için, paralelleştirme işlemi için mpiJava ve mpich kütüphaneleri kullanılmıştır. mpiJava, standart MPI(Message Passing Interface) için geliştirilen nesneye dayalı Java arayüzüdür [11,12,13,14]. mpi komutlarının kullanılması ve programın gerekli yerlerine eklenen mpi rutinleri sayesinde, programın paralelleştirilmesi sağlanmıştır.

Parallelleştirme, sınama ya da öğrenme kümesinin dağıtılması ile sağlanabilirdi. İlk durumda, öğrenme kümesinde herhangi bir değişiklik yapmaksızın sınama kümesinin belirlenen işlemci sayısına bölünmesini içerirken, ikinci durumda ise sınama kümesinde değişiklik yapmadan, öğrenme kümesinin işlemci sayısına bölünmesi ile gerçekleştirilebilir. Bu çalışmada, ilk durum ele alınarak, sınama kümesinde bulunan 2500 örnek, işlemci sayısına göre bölünerek, herbir kısmı belirlenen bilgisayara gönderilmiştir. Çalışma için, özellikleri birebir aynı 4 makine kullanılmıştır. Bilgisayarlarda işlenen veriler mpi rutinleri kullanılarak birleştirilmiş ve sonuçların bulunması sağlanmıştır.

4. DENEYSEL SONUÇLAR

Doğruluk oranının değerlendirilmesi ile ilgili yapılan deneylerde, sınama kümesinin bir kısmı kullanılarak %80 doğruluk oranı ve %80 f-ölçütü değerleri elde edilirken, tüm sınama kümesi kullanılarak elde edilen sonuçlarda %70 doğruluk ve %75 f-ölçütü değerleri bulunmuştur. Bu değerler, 3 ayrı sınama kümesine göre değişiklik göstermektedir. Deneysel çalışmalarda yukarıdaki bölümlerde de bahsedildiği gibi k'nın farklı değerleri ve farklı eşik değerleri için sonuçlar bulunmuştur. Aşağıda, k değeri 5 ve eşik değeri 70000 iken üretilen sonuç gösterilmektedir.

Doğruluk:0.70
Hata oranı:0.294
Kesinlik:0.6487579433853264
Anma:0.8984
F-olcutu:0.7534384434753439

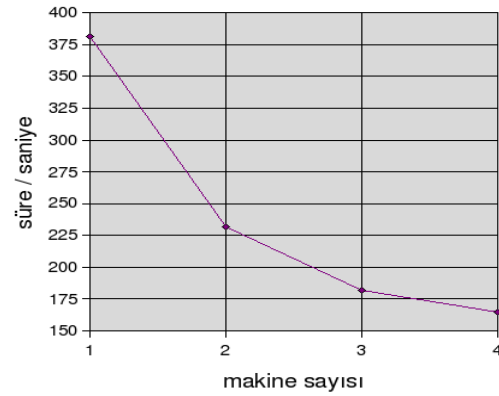
Elde edilen sonuç, çok iyi olmasa da, iyileştirme algoritmaları ile oranın biraz daha yükselmesi sağlanabilir.

Zamanın değerlendirilmesi ilgili yapılan çalışmalarda ise, paralelleştirme yapılmadan ve yapıldıktan sonraki durumlar göz önüne alınarak sonuçlar karşılaştırılmıştır. Deney ortamımızda sadece dört makine kullanarak elde edilen kazançlar gözlemlenmiştir. Aşağıdaki Tablo 4 ve Şekil 1' de görüldüğü gibi harcanan zaman ile kullanılan makine sayısı ters logaritmik bir ilişki içersindedir.

Tablo 4. Makine sayısına göre değişen süre

Makine sayısı	Süre Saniye
1	381
2	232
3	182
4	165

KNN



Şekil 1. Makine sayısına göre azalan süre grafiği

Makine sayısı arttıkça algoritmanın harcadığı zaman azalır. Ama belli bir makine sayısına gelindiğinde bu azalma duracaktır. Bu deneyi daha fazla makine ile tekrarlayıp kullanılacak maksimum makine sayısını hesaplamak gerekir. Bu deneme sonraki araştırmalarımız arasındadır.

5. SONUÇ

Çalışmada istenmeyen elektronik iletilerin belirlenmesi için kişisel bir filtreleme modeli geliştirilmeye çalışılmıştır. 2006 yılında, ECML/PKDD tarafından düzenlenen yarışmadaki veri kümeleri kullanılarak, KNN algoritması gerçekleştirilmiştir. Yapılan işlemin büyüklüğünden ötürü, geçirme veri kümesi üzerinde k ve eşik değerleri belirlenmiştir. Buna göre, en uygun k ve eşik değeri atanarak sonuçlar, etiketli veriler ile karşılaştırılmıştır. Sonuçların kısa sürede alınması için algoritma paralelleştirilmiştir.

Deneysel sonuçlara göre, %70 oranında doğruluk oranına ulaşılmıştır. Çok iyi bir sonuç olmasa da iyileştirme algoritmaları ile oranın biraz daha yükselmesi sağlanabilir. Bu da sonraki çalışma planımız içindedir. Bununla beraber daha fazla makine kullanılması da ilerki çalışma planımız arasındadır.

6. KAYNAKLAR

- [1] T. Fawcett, "In vivo" Spam Filtering: A challenge problem for data mining", KDD Explorations vol.5 no.2, Dec 2003. pp.140-148
- [2] X. Carreras and L. Marquez. Boosting trees for antispam email filtering. In Proceedings of RANLP-2001, 4th International Conference on Recent Advances in Natural Language Processing, 2001.
- [3] J. Provost. Naive-bayes vs. rule-learning in classification of email. Technical Report AI-TR-99-284, University of Texas at Austin, Artificial Intelligence Lab, 1999.
- [4] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. A bayesian approach to filtering junk e-mail. Madison, Wisconsin, 1998. AAAI Technical Report WS-98-05.
- [5] K. Schneider. A comparison of event models for naive bayes anti-spam e-mail filtering. In Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03), 2003.
- [6] H. Katirai. Filtering junk e-mail: A performance comparison between genetic programming & naive bayes. 1999.
- [7] A. Kolcz and J. Alsepector. SVM-based filtering of email spam with content-specific misclassification costs. TextDM'2001 ICDM-2001 Workshop on Text Mining, San Jose, CA, 2001.
- [8] Data Mining Cup 2003. <http://www.data-mining-cup.com/2003/Wettbewerb/1059704704/>, 2003.
- [9] Discovery Challenge, <http://www.ecmlpkdd2006.org/challenge.html>, 2006
- [10] K Nearest Neighbors Tutorial people. revoledu.com/kardi/tutorial/KNN/HowTo_KNN.html
- [11] The HPJava Project, <http://www.hpjava.org/mpiJava.html>
- [12] MPICH-A Portable Implementation of MPI, <http://www-unix.mcs.anl.gov/mpi/mpich1/>
- [13] mpiJava Example Programmes, <http://users.cs.cf.ac.uk/David.W.Walker/CM0323/code.html>
- [14] P2P-MPI Entities, <http://grid.u-strasbg.fr/p2pmpi/documentation/samples.html>