GU J Sci, Part C, 6(3): 571-581 (2018)

Gazi Üniversitesi

Fen Bilimleri Dergisi

PART C: TASARIM VE TEKNOLOJİ

dergipark.gov.tr/http-gujsc-gazi-edu-tr

Büyük ölçekli veri setleri için GPU hızlandırmalı melez bir GA-DVM: Cu-GA-DVM

Musa PEKER[1], Osman ÖZKARACA[1,*]

[1]*Muğla Sıtkı Koçman Üniversitesi, Teknoloji Fakültesi, Bilişim Sistemleri Mühendisliği Bölümü, 48000, Menteşe/MUĞLA*

Makale Bilgisi

Başvuru: 01/02/2018
Düzeltme: 25/04/2018
Kabul: 31/05/2018

Anahtar Kelimeler

CUDA
Genetik Algoritma
DVM
Melez algoritma

Keywords

CUDA
Genetic Algorithm
SVM
Hybrid Algorithm

Öz

Bu çalışmada Genetik Algoritma (GA) ve Destek Vektör Makinelerinden (DVM) oluşan melez bir yöntemin CUDA (Compute Unified Device Architecture-Birleşik Hesaplama Aygıt Mimarisi) tabanlı hız optimizasyonu gerçekleştirilmiştir. Makine öğrenmesinde, geliştirilen yöntemlerin yüksek doğruluk değerlerinde başarı vermesi hedeflenir. Ayrıca önerilen algoritmanın sonuçları bulurken hızlı bir şekilde çalışması da yine hedeflenen bir durumdur. Bu çalışmada, özellikle gerçek zamanlı uygulamalarda önemli bir parametre olan hız parametresi dikkate alınmakta ve verilerin hızlı bir şekilde sınıflandırılması için yeni bir GPU (Graphic Processing Unit-Grafik İşlemci Birimi) teknolojisi kullanılmaktadır. Bunun için grafik işlemciler üzerinde programlama yapmamızı sağlayan CUDA programlamadan yararlanılmıştır. Sınıflandırma algoritması olarak genetik algoritmayla optimize edilmiş destek vektör makinesi kullanılmıştır. Deneyler 384 CUDA çekirdeğinden oluşan NVIDIA GeForce 940MX ekran kartına sahip bir bilgisayar üzerinde gerçekleştirilmiştir. Büyük ölçekli veri kümeleri üzerinde yapılan deneylerde, CUDA programlamanın sonuçlar üzerinde pozitif etkilerinin olduğu görülmüştür. Bu şekilde makine öğrenmesi uygulamalarında sınıflandırma aşamasında grafik işlemciler ile gerçek zamanlı uygulamalar için hızlı bir sistemin altyapısı oluşturulabilir.

A GPU accelerated hybrid GA-SVM for large scale datasets: Cu-GA-SVM

Abstract

In this study, CUDA based speed optimization of a hybrid method consisting of Genetic Algorithm and Support Vector Machines has been performed. In machine learning, it is aimed to achieve high accuracy values from the developed methods. It is also a target for the proposed algorithm to work quickly while finding the results. In this study, speed parameter which is indispensable especially in real time applications is taken into consideration and a new GPU technology is used to classify the data quickly. Therefore, CUDA programming, which allows us to program on graphics processors of which importance and use are increasing in recent years, has been benefited from. Support vector machine optimized by genetic algorithm has been used as the classification algorithm. The experiments have been performed on a computer with NVIDIA GeForce 940MX graphics card, which consists of 384 CUDA core. Experiments performed on large scale data sets have shown that CUDA programming has positive effects on the results. In this way, the infrastructure of a quick system for real-time applications can be created by using the graphics processors in the classification phase of the machine learning applications.

1. GİRİŞ (INTRODUCTION)

Teknolojideki gelişmelere bağlı olarak, kullanılan veri setleri çok büyük boyutlara ulaşmış ve gün geçtikçe büyümeye devam etmektedir. Bu verileri analiz etmek için çoğunlukla makine öğrenmesi yöntemleri kullanılmaktadır. Büyüyen veri kümelerinin analizini, makine öğrenmesi yöntemleri ile makul bir sürede tamamlamak için özellikle de son yıllarda GPU programlama teknolojilerinin önem kazandığı görülmektedir.

*İletişim yazarı, e-mail: osmanozkaraca@mu.edu.tr

DOI: 10.29109/gujsc.388244

Son yıllarda özellikle de NVIDIA firmasının GPU programlama için ciddi yatırımlar yaptığını söylemek mümkündür. Geliştirilen teknolojiler ile yüksek hesaplama güçlerine ulaşılmaktadır. Örneğin şu anda bazı NVIDIA grafik kartları toplamda 1790 GFlops hesaplama gücü sağlayan 2 GPU içermektedir. Intel i7 dört çekirdekli işlemcisi için bu değer yaklaşık 50 GFlops'dur.

Literatürde veri madenciliği algoritmalarının hızlandırılmasına yönelik sunulmuş çalışmalar mevcuttur. Lo ve arkadaşları [1], bir paralel karar ağacı önerdiler. Geliştirilen algoritmada GPU hesaplamalar için, CPU ise akış kontrolü için kullanılmıştır. Sonuç olarak, önerilen paralel karar ağacı geleneksel karar ağacından 5~55 kat daha hızlıdır. Xavier ve arkadaşları [2], BLAS'ın (Basic Linear Cebir Subprogramı) bir CUDA uygulaması olan CUBLAS'ı kullanarak sinir ağının geri yayılımının paralel eğitimini anlatmaktadır. Farklı gizli nöron sayılarına göre CPU ve CUDA uygulamalarını karşılaştırdılar ve klasik bir CPU üzerinde seri yürütmeye karşı GPU'nun performansını bildirdiler. Bhimani ve arkadaşları [3], k-ortalamalar kümeleme algoritmasının MPI, OpenMP ve CUDA üzerinde paralel uygulanmasını geliştirdi. Sonuç olarak küçük veriler için OpenMP en iyi performansı gösterirken; daha büyük veriler için CUDA en iyi hızlanmayı sağlamıştır. Zhang ve arkadaşları [4], büyük veri kümesindeki k-ortalamalar kümeleme algoritmasını paralel hale getirmek için MPI kullanarak bir çözüm önerdi. Catanzaro ve arkadaşları [5] GPU kullanarak ikili bir DVM sınıflandırıcıyı eğitmek için bir yöntem önermişlerdir. Map-reduce tekniğinin kullanıldığı çalışmada, Libsvm yazılımıyla karşılaştırıldığında belirgin hız artışı bildirilmiştir. Cao ve arkadaşları [6], bir küme sisteminde MPI ile uygulanan bir Paralel Sıralı Minimal Optimizasyon algoritması geliştirmiştir. Ting He ve arkadaşları [7], CUDA işlevlerini kullanarak sinir ağlarını eğitmek için GPU standart yeteneklerini kullandılar ve GPU üzerinde matris çarpımlarının ve vektör işlemlerinin çoğunu yaparak CPU'ya kıyasla 5.21X'lik bir hızlanma faktörü elde ettiler. Wang ve arkadaşları [8], en yakın komşu bölümleme (NNP) yönteminin CUDA tabanlı optimizasyonunu gerçekleştirdiler. NNP yöntemi, geleneksel sinir ağ sınıflandırıcılarını geliştirmek için kullanılan yüksek performanslı bir yaklaşımdır. Bununla birlikte, NNP modelinin yapım süreci, özellikle büyük veri setleri için çok zaman alan bir işlemdir ve böylece uygulama alanını sınırlandırmaktadır. CUDA tabanlı optimizasyon ile performans açısından iyi sonuçlar elde edilmiştir. Krawczyk [9], dengesiz veri setlerinin sınıflandırılması için aşırı öğrenme makinesini kullanmış ve hız optimizasyonu için CUDA kullanmıştır. Farklı veri setleri üzerinde yapılan çalışmada hız açısından ortalama 10X başarı elde edilmiştir. Singh ve arkadaşları [10], GPU tabanlı bir k-en yakın komşu algoritması geliştirdi. Farklı veri setleri üzerinde yapılan çalışmada hız açısından ortalama 10X başarı elde edilmiştir. Arslan ve arkadaşları [11], büyük veri kümeleri için GPU tabanlı paralel bir DVM algoritması geliştirdi. Önerilen yöntem ile hız performansı açısından etkili sonuçlar elde edilmiştir. Ayrıca sezgisel optimizasyon algoritmalarının [12, 13] hızlandırılmasında ve dalgacık dönüşümü [14-16] gibi önemli algoritmaların hızlandırılmasında da CUDA'nın kullanıldığı ve başarılı sonuçların elde edildiği görülmektedir.

Bu çalışmada bir GPU programlama türü olan CUDA ile genetik algoritma ve destek vektör makinelerinden oluşan melez bir yöntemin hızlandırılması amaçlanmıştır. Makalenin organizasyonu şu şekildedir: Bölüm 2'de bu çalışmada kullanılan metotlar hakkında bilgiler sunulmuştur. Bölüm 3'te metotların hızlandırılmasına yönelik CUDA tabanlı çözüm önerileri ve uygulamalara yer verilmiştir. Bölüm 4'te elde edilen sonuçlar hakkında bilgiler verilmiştir.

2. MATERYAL VE METOT (MATERIAL AND METHOD)

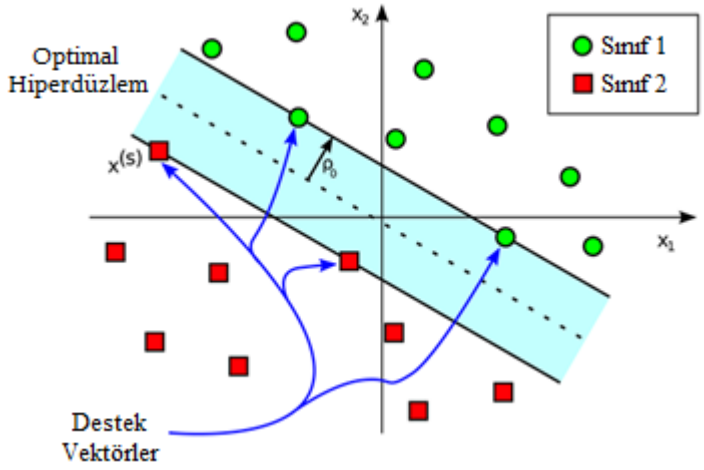
2.1. Destek Vektör Makinesi Algoritması (Support Vector Machine Algorithm)

DVM algoritması, sınıflandırma ve regresyon problemlerinin çözümü için geliştirilen bir algoritmadır. Bu algoritma Lagrange çarpanları denklemlerini temel almaktadır. Bu algoritma, veri noktalarını en iyi şekilde farklı sınıflara ayıran optimum ayırıcı düzlemin bulunmasını hedeflemektedir. Destek vektör makinelerinin, doğrusal ve doğrusal olmayan iki durumu söz konusudur. Doğrusal DVM, doğrusal olarak ayırt edilebilen problemlere uygulanmaktadır. Doğrusal DVM yapısı Şekil 1(a)'da verilmiştir. $(y_1, y_2, \dots, y_n)$ veri seti, $z_i \epsilon (-1, 1)$ sınıf etiketleri ve b eşik değeri olsun. DVM algoritmasında veri, $g(x) = w^T y + b = 0$ hiper düzlemi ile ayrılmaktadır.

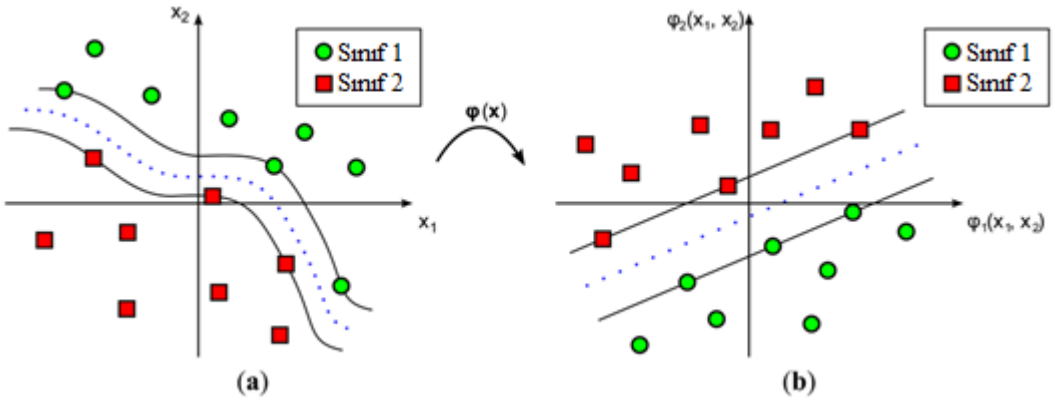
$$w^T y_t + b \geq +1, \qquad \text{eğer } z_t = +1 \text{ (sınıf 2)} \tag{2.1}$$

$$w^T y_t + b \leq -1, \qquad \text{eğer } z_t = -1 \text{ (sınıf 1)} \tag{2.2}$$

$g(x) = w^T y + b = 0$ hiper düzleminin alt ve üst tarafında kalan noktalar Denklem 2.1 ve 2.2 kullanılarak hesaplanmaktadır.



(a)



(b)

Şekil 1. *DVM'nin geometrik sunumu (a) Doğrusal DVM (b) Doğrusal olmayan DVM [17]*

Doğrusal olarak ayırt edilme durumu olmadığında verinin daha yüksek boyutlu bir uzaya taşınması bir çözüm olarak düşünülebilir. Doğrusal olmayan destek vektör makinelerinin dayandığı temel teori budur. DVM bu işlemleri gerçekleştirmek için çekirdek fonksiyonlarını kullanmaktadır. Bu durumda a boyutlu bir veri kümesi, $b > a$ olacak şekilde b boyutlu yeni bir veri kümesine taşınmaktadır. Bu DVM yapısı Şekil 1(b)'de sunulmuştur. DVM için geliştirilen çok sayıda çekirdek fonksiyonu bulunmaktadır. Bu çalışmada radyal tabanlı çekirdek fonksiyonu kullanılmıştır. Bu fonksiyonun denklemi denklem 2.3'te sunulmuştur.

$$K(y_i, y_j) = e^{-\frac{\|y_i - y_j\|^2}{2\sigma^2}} \qquad 2.3$$

2.2. Genetik Algoritma (Genetic Algorithm)

GA, optimizasyon problemlerinde kullanılan sezgisel bir algoritmadır. Bu algoritma mutasyon, kalıtım, çaprazlama ve seçim gibi evrimsel biyolojideki tekniklerden ilham alınarak geliştirilmiştir. Genetik algoritma, arama uzayı büyük olan ve ayrıca değişken sayısı çok fazla olan, çok boyutlu problemlerde başarılı sonuçlar verebilmektedir. Genetik algoritmanın işlem adımları aşağıda verilmiştir [18].

1. Başlangıç popülasyonunu rastgele olarak üret.
2. Popülasyon içindeki tüm kromozomların amaç fonksiyonu değerlerini hesapla.

3. Tekrar üreme, çaprazlama ve mutasyon operatörlerini uygula.
4. Oluşturulan her yeni kromozomun amaç fonksiyonu değerlerini bul.
5. Amaç fonksiyonu değerleri kötü olan kromozomlar popülasyondan çıkar.
6. 3-5 arasındaki adımlar tekrar et.

2.3. Melez GA-DVM algoritması (Hybrid GA-SVM algorithm)

Etkili bir DVM modeli oluşturmak için, modelin parametrelerinin (C ve γ) önceden seçilmesi gerekir. C parametresi, eğitim hatasını asgariye indirme ve DVM modelinin karmaşıklığı arasındaki ödünleşim (tradeoff) maliyetini belirler. Daha büyük bir C değeri ile, eğitim örneğinin tahmini doğruluğu daha yüksektir. Bununla birlikte, bu durum fazla eğitim problemine neden olabilir. RBF çekirdek fonksiyonunun γ parametresi, girdi alanından yüksek boyutlu öznitelik alanına kadar doğrusal olmayan bir haritalama tanımlar. γ değeri RBF fonksiyonunun şeklini etkiler. Dolayısıyla, parametreler (C ve γ), DVM modelinin verimlilik ve genelleme performansı üzerinde güçlü bir etkiye sahiptir. Günümüzde, parametrelerin seçimi esas olarak deneyimlere bağlı olup olgun teori yönlendirmesinden yoksundur. Literatürde ağırlıklı olarak, Grid (ızgara sistem) arama tekniği kullanılmaktadır. Bununla birlikte, grid algoritması zaman almaktadır.

Farklı alanlardaki ilgili araştırmalara göre, GA'nın parametrelerin belirlenmesinde daha iyi bir seçim olduğu kanıtlanmıştır [19, 20]. Bu şekilde, DVM modelinin tahmin performansını artırabilir. Bu nedenle, bu çalışmada DVM modelinin en uygun parametreleri için GA tercih edildi. Bu algoritmada uygunluk fonksiyonu olarak ortalama karesel hata değeri kullanılmıştır. GA-DVM yönteminin kaba kodu, Algoritma 1'de görülebilir.

Algoritma 1. GA, DVM parametrelerini aşağıdaki adımlarla optimize eder.

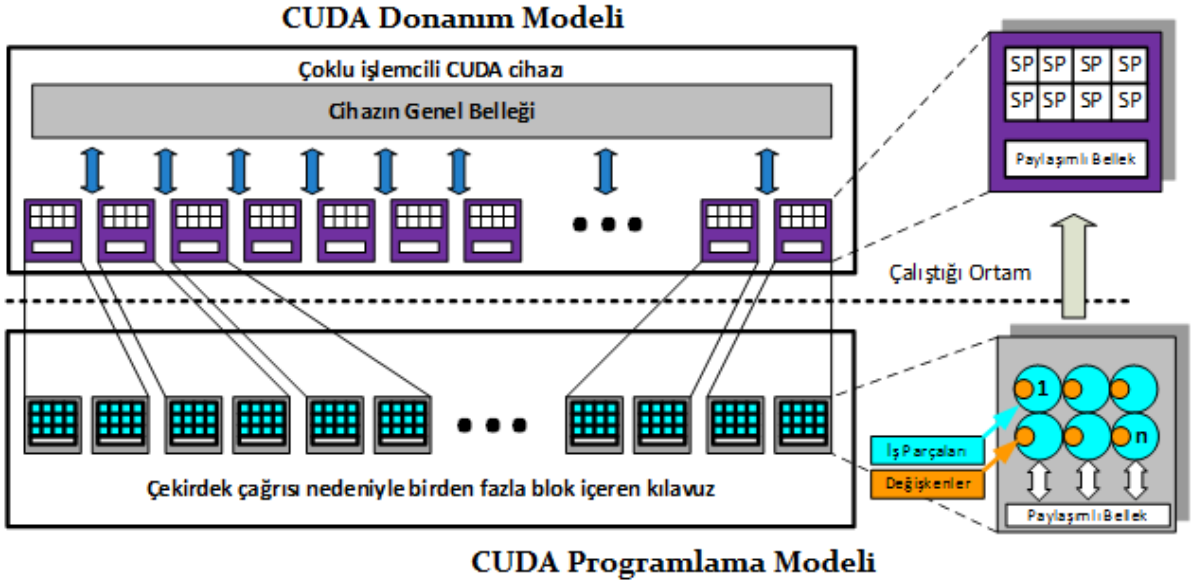
Popülasyon büyüklüğü ve iterasyon sayısı gibi GA'nın başlangıç parametrelerini belirleyin.
C ve γ parametrelerinin kodlama aralığını belirleyin. Kromozom üretmek için gerçek sayı kodlaması tercih edildi.
Ortalama Karesel Hata (MSE), uygunluk fonksiyonu (fitness function) olarak belirlendi.
repeat
Rulet seçimi (Roulette selection), elit strateji (elite strategy) ile işbirliği içinde kullanılır.
Çapraz (crossover) ve mutasyon (mutation) operatörleri, bir çocuk popülasyonu (child population)oluşturmak için kullanılır.
if uygunluk kabulse **then**
    En iyi bireysel ve optimal çözümü üret
else
    Seçme, çapraz ve mutasyon operatörlerini çalıştır
end if
until (durdurmu kriteri sağlandı mı?)

2.4. CUDA Programlama (CUDA Programming)

Bu bölümde CUDA programlama ve donanım modellerine genel bir bakış sunulmaktadır. CUDA paralel programlama; standart programlama diline paralellik katarak, bir uygulamanın birden fazla işlemci çekirdeği ve eş zamanlı iş parçacıkları üzerinde çalıştırılabilmesini sağlar. CUDA programlama hakkında daha fazla ayrıntı için lütfen (NVIDIA [21]) kaynağına bakılabilir. Şekil 2, bir yazılım CUDA bloğunu bir donanım CUDA çoklu işlemcisine eşleme yapan CUDA programlama modelini tasvir etmektedir. Çoklu işlemciye bir dizi öbek atanabilir ve bunlar CUDA programlama ortamı tarafından dâhili olarak zaman paylaşımlıdır. Her çoklu işlemci, CUDA cihazının çözgü (warp) boyutuna dayalı olarak, bir öbek (block) içinde mevcut iş parçacıklarını bir zaman paylaşımıyla çalıştıran bir dizi işlemciden oluşur. Her çoklu işlemcide ayrıca küçük bir paylaşımlı bellek (shared memory), bir grup 32-bit saklayıcı, doku (texture) ve tüm işlemcilerin içerisinde bulunan sabit hafıza (constant memory) ön bellekleri bulunur. Çoklu işlemcideki işlemciler, aynı komutları istediğiniz zaman farklı veriler üzerinde yürütür. Bu, CUDA'yı bir SIMD modeli yapar. Çoklu işlemciler arasındaki iletişim, çoklu işlemci içindeki tüm işlemcilerin erişebildiği cihaz genel belleği yoluyla olur. Bir öbeğin iş parçacıkları arasında senkronizasyon mümkündür. Öbekler arasında senkronizasyon yalnızca çekirdek sınırlarında mümkündür.

CUDA API (Uygulama Programlama Arayüzü), C dilinin bir uzantısı olarak kodlanabilen bir dizi kütüphane fonksiyonu sağlar. Bir derleyici CUDA cihazı için çalıştırılabilir kod oluşturur. CPU bir CUDA cihazını çok çekirdekli eş işlemci olarak görür. Kod, CUDA işlemcilerinde zaman paylaşımlı olarak, iş parçacıkları paralel olarak çözgü boyutu yığınlarında çalışırken yürütür. Her iş parçacığı, hesaplaması için bir dizi özel kayıt kullanabilir. Bir iş parçacığı yığını (öbek adı verilir) belirli bir zamanda bir çoklu işlemcide çalışır. Her öbeğin iş parçacıkları, az miktardaki ortak paylaşılan hafızaya erişebilir. Senkronizasyon engelleri, bir öbeğin tüm iş parçacıkları için mevcuttur. Bir öbekler grubu, tek bir çoklu işlemciye atanabilir ancak bunların yürütülmesi zaman paylaşımlıdır. Mevcut paylaşılan bellek ve kayıtlar, çoklu işlemciyi zaman paylaşımlayan tüm öbekler arasında eşit olarak bölünür. Bir cihazdaki yürütme, yığın olarak grid/ızgara olarak bilinen bir takım öbekler oluşturur.

Her bir iş parçacığı, çekirdek adı verilen tek bir komut kümesini çalıştırır. İş parçacıkları ve bloklara, çalışırken iş parçacığında erişilebilecek benzersiz bir kimlik verilir. Bunlar, bir iş parçacığının çekirdek görevini verinin bir parçası üzerinde gerçekleştirmek için SIMD yürütme ile sonuçlanacak şekilde kullanılabilir. Algoritmalar, genel bellek vasıtasıyla veri paylaşan ve her bir çekirdek sonunda yürütümlerini senkronize eden çoklu çekirdekleri kullanabilir.



Şekil 2. *Çok işlemcili haritalama blokunu gösteren CUDA donanım modeli (üstte) ve programlama modeli (altta).*

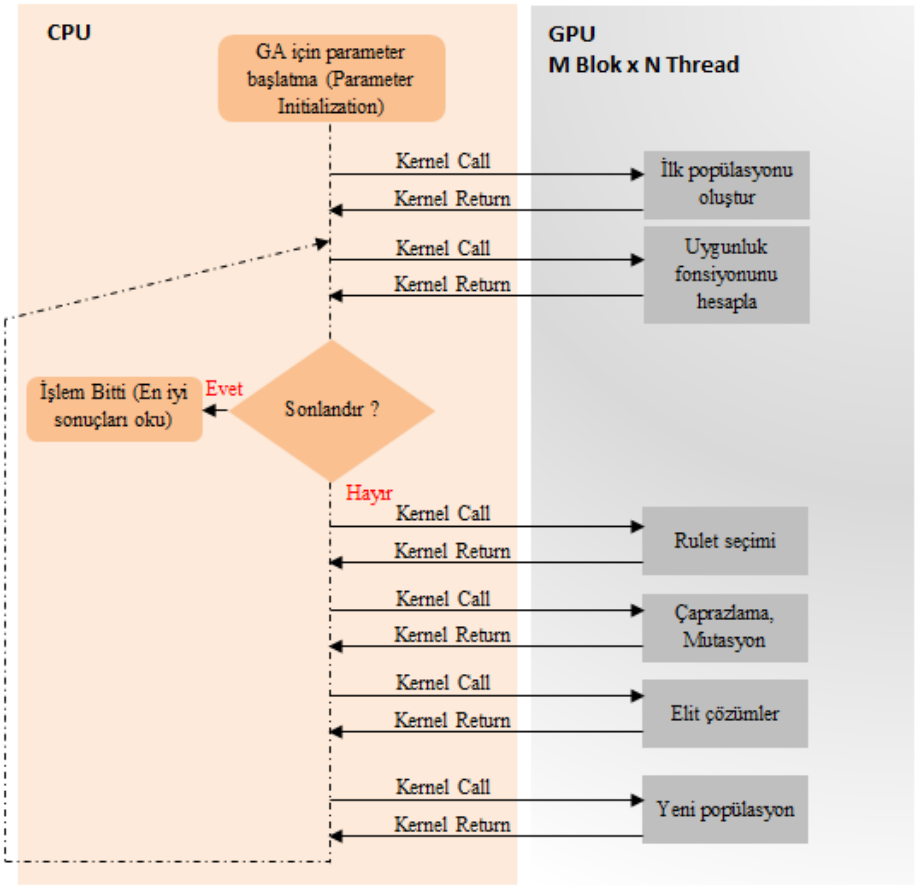
Bu çalışmada CUDA programlama için NVIDIA GeForce 940MX grafik kartı kullanılmıştır. Bu grafik kartında 384 CUDA çekirdeği vardır. CPU performansını değerlendirme için aynı bilgisayarın işlemcisi kullanılmıştır. Bu işlemci Intel (R) Core (TM) i7-7500U, 2.70 GHz ve 8 GB RAM özelliklerindedir.

Çalışma kapsamında birçok çekirdek fonksiyonu yazılmıştır. Bunun yanında performans optimizasyonu için bir takım optimize edilmiş CUDA kütüphaneleri de kullanılmıştır. Bu kapsamda cuBLAS kütüphanesi kullanılmıştır. NVIDIA cuBLAS kütüphanesi, standart temel doğrusal cebir altyordamlarının (BLAS) hızlı bir GPU hızlandırılmış uygulamasıdır.

3. GA-DVM ALGORİTMASININ CUDA TABANLI GERÇEKLEŞTİRİMİ (CUDA-BASED IMPLEMENTATION OF GA-SVM ALGORITHM)

Gerçekleştirilen çalışmada Matlab, C, CUDA bir arada çalışmaktadır. Paralel programlama bu aşamada matris-vektör çarpımlarının hesaplama karmaşıklığını düşürmek için kullanılmıştır. Matris oluşturulması, vektör-vektör atama ve çarpım gibi işlemlerin tümü hesaplama karmaşıklığına neden olur. Bu noktada CUDA ile bu işlemlerin hızlandırılması sağlanmıştır. Gerçekleştirilen çalışmada, CPU üzerinde sıralı çalışan DVM algoritması içerisinde kullanılan öğrenme ve test fonksiyonları paralel hale getirilmiştir.

Sistem, GPU'da en az hafıza alanı kullanacak şekilde tasarlanmıştır. GPU optimizasyonu yapılarak paralelleştirilen GA algoritması işlemi Şekil 3'de verilmiştir.



Şekil 3. *GPU Destekli GA Algoritması*

GPU optimizasyonu yapılarak paralelleştirilen DVM sınıflandırma işlemi ise Şekil 4'de verilmiştir. Şekilde çekirdek fonksiyonun hesaplanmasını hızlandırmak için kullanılan CUDA çekirdek fonksiyonu da verilmiştir.

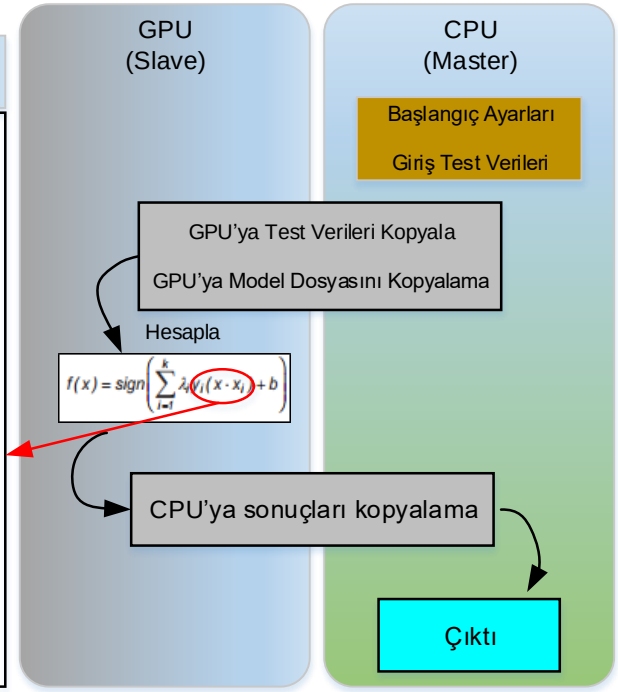
RBF çekirdek fonksiyonunun CUDA tabanlı hesaplanması

```
_device_ cudafloat RBFKernelCUDA (int a0, int a1, cudafloat *
    samples, int n_samples, int num_dimensions, cudafloat *
    kernel_args)
{

cudafloat  sum_variable=0;

for (int i=0; i< num_dimensions;i++) {

cudafloat n0_i=samples [n_samples*a0+i];
cudafloat n1_i=samples [n_samples*a1+i];
…
cudafloat _difference= n0_i - n1_i;
cudafloat _difference_sq= _difference* _difference;
sum_variable+=_difference_sq;
}
…
cudafloat a=kernel_args[0];
…
return cuda_pow(e, -1*(sum_variable/2*a*a)
….
}
```



Şekil 4. *GPU destekli DVM Sınıflandırması*

3.1. CUDA Uygulamaları ve Deneysel Sonuçlar (CUDA Applications and Experimental Results)

Bu çalışmada geliştirilen CUDA yazılımı ile CPU üzerinde çalışan LIBSVM yazılımları ile elde edilen sonuçlar karşılaştırılmıştır. Deneyler büyük veri setleri üzerinde gerçekleştirilmiştir. Bu veriler UCI makine öğrenme veri setleri sitesinden alınmıştır. Deneylerde çekirdek fonksiyonu olarak radyal tabanlı çekirdek fonksiyonu kullanılmıştır. Çalışmada kullanılan veri setleri hakkında bilgiler Tablo 1'de sunulmaktadır.

Tablo 1. *Veri setleri ve özellikleri*

Veri Seti	Eğitim Örnekleri Sayısı	Test Örnekleri Sayısı	Öznitelik Sayısı	Sınıf Sayısı
ADULT	32651	16281	123	2
WEB	49749	14951	300	2
MNIST	60000	10000	780	10
USPS	7291	2007	256	10

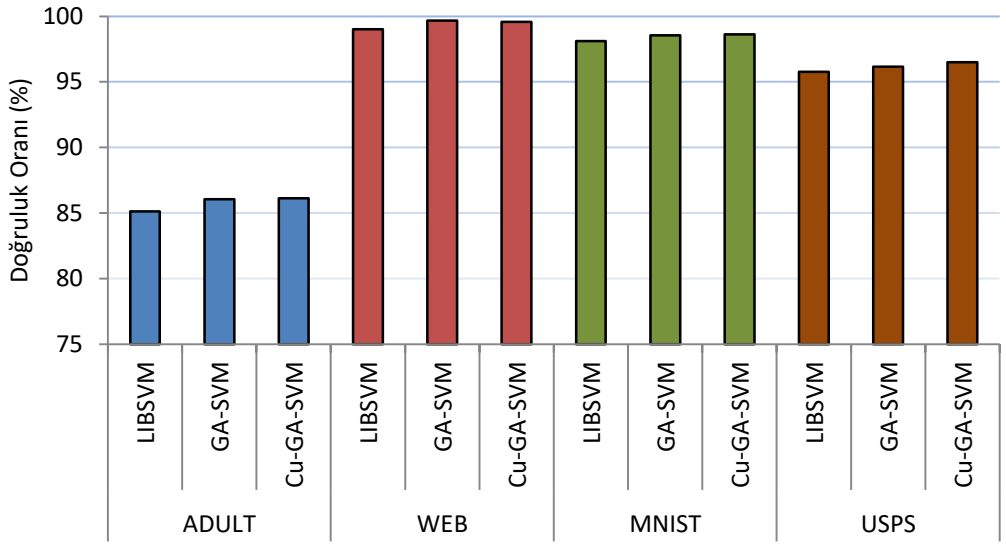
Bu veri setleri üzerinde yapılan deneyler neticesinde elde edilen sonuçlar Tablo 2 ve Tablo 3'te sunulmaktadır. Sonuçlar, 10 farklı deney sonucu elde edilen değerlerin ortalamasından oluşmaktadır. Tablo 2'de farklı veri setleri için elde edilen doğruluk oranları sunulmaktadır. Bu testte her üç yöntem de, tüm veri kümelerinde birbirine çok yakın bir performans sergilemektedir.

Tablo 2. *Farklı DVM algoritmaları ile 4 farklı veri kümesi için elde edilen doğruluk oranları*

Veri seti	DVM	Eğitim Doğruluğu	Test Doğruluğu
ADULT	LIBSVM	85.85%	85.12%
	GA-DVM	86.38%	86.05%
	Cu-GA-DVM	86.24%	86.12%
WEB	LIBSVM	99.51%	99.02%

	GA-DVM	99.75%	99.68%
	Cu-GA-DVM	99.66%	99.57%
MNIST	LIBSVM	99.60%	98.12%
	GA-DVM	99.88%	98.56%
	Cu-GA-DVM	99.88%	98.61%
USPS	LIBSVM	99.98%	95.77%
	GA-DVM	99.99%	96.15%
	Cu-GA-DVM	99.99%	96.50%

Sonuçların grafiksel sunumu aşağıda Şekil 5'de verilmiştir.



Şekil 5. *Farklı DVM algoritmaları ile 4 farklı veri kümesi için elde edilen doğruluk oranları*

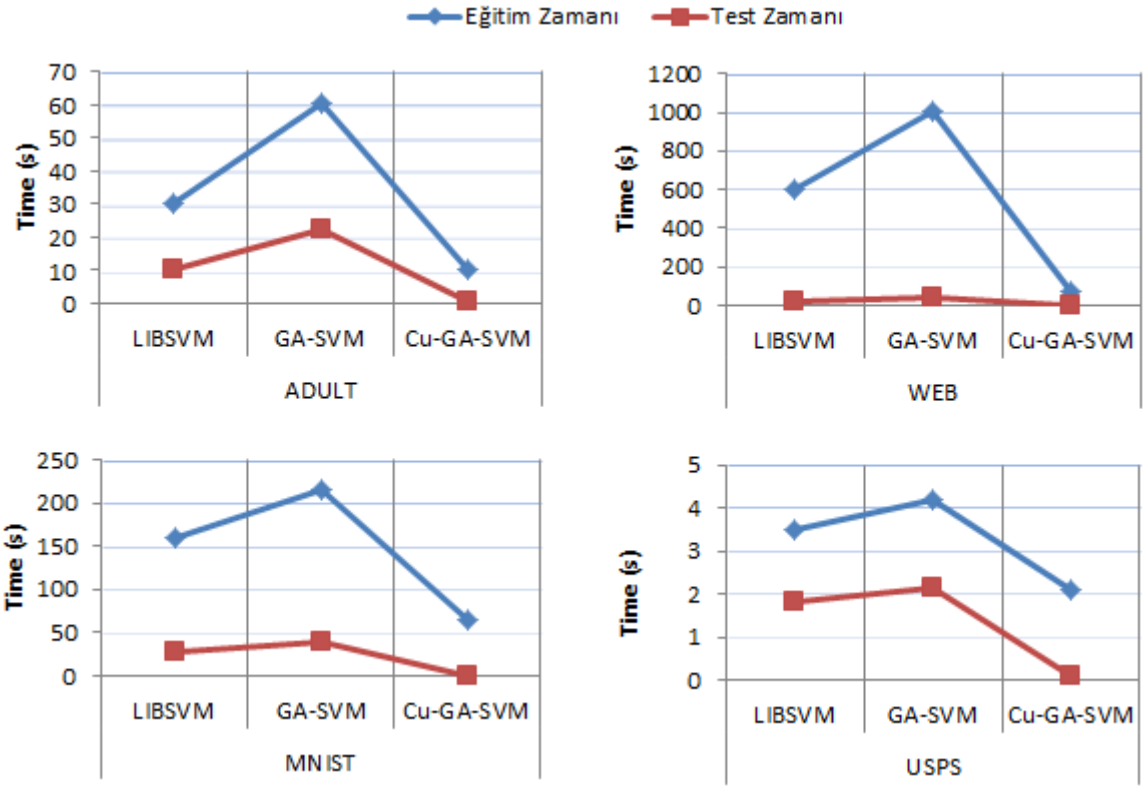
Hız parametresi dikkate alınarak gerçekleştirilen deney sonuçları ise Tablo 3'de verilmiştir.

Tablo 3. *DVM algoritması için GPU-CPU hız karşılaştırması*

Veri Seti	DVM	Eğitim Zamanı (s)	Hızlanma (Speed Up)	Test Zamanı (s)	Hızlanma (Speed Up)
ADULT	LIBSVM	30.50		10.11	
	GA-DVM	60.22		22.14	
	Cu-GA-DVM	10.16	5.92X	0.745	30.52X
WEB	LIBSVM	600.42		20.26	
	GA-DVM	1005.10		35.16	
	Cu-GA-DVM	66.28	15.16X	0.84	41.85X
MNIST	LIBSVM	160.12		27.85	
	GA-DVM	215.60		38.22	
	Cu-GA-DVM	65.05	3.31X	0.64	45.5X
USPS	LIBSVM	3.5		1.82	
	GA-DVM	4.2		2.16	
	Cu-GA-DVM	2.1	2X	0.08	27X

Tablo 3'den de görüleceği gibi GA-DVM algoritmasında CUDA ile ciddi oranda hızlanma sağlanmıştır. Eğitim verileri üzerinde gerçekleştirilen deney sonuçları dikkate alındığında; ADULT veri setinde 5.92X, WEB veri setinde 15.16X, MNIST veri setinde 3.31X ve USPS veri setinde 2X hızlanma sağlanmıştır.

Test verileri üzerinde gerçekleştirilen deney sonuçları dikkate alındığında ise; ADULT veri setinde 30.52X, WEB veri setinde 41.85X, MNIST veri setinde 45.5X ve USPS veri setinde 27X hızlanma sağlanmıştır. Elde edilen sonuçların grafiksel sunumu ise Şekil 6'da verilmiştir.



Şekil 6. *4 farklı veri kümesinin eğitim ve test verileri için harcanan zaman değerleri*

Tablo 1 ve Tablo 3'deki veriler incelendiğinde, veri setindeki veri sayısı arttıkça CUDA'nın olumlu etkisinin de arttığı görülmektedir. Örneğin en büyük örnek sayısına sahip olan MNIST veri setinde 45.5X'lik bir başarı elde edilmişken, en düşük örnek sayısına sahip olan USPS veri setinde 27X'lik bir başarı elde edilmiştir. Genel olarak veri boyutu arttıkça CUDA programlamanın hız optimizasyonu açısından olumlu etkisi de artmaktadır.

Cu-GA-DVM algoritmasının başarılı sonuçlar vermesinin nedenleri arasında yapılan çeşitli optimizasyonlar bulunmaktadır. Yapılan optimizasyon çalışmalarından bazıları aşağıda sunulmuştur.

1. Paralelleştirilebilen kod parçalarının CUDA üzerinde gerçekleştirilmesine dikkat edilmiştir. Seri olması gereken işlemler ise CPU üzerinde çalıştırılmıştır.
2. CUDA ile işlenecek verilerin bir seferde GPU üzerine alınması ve GPU üzerinde tüm işlemler yapıldıktan sonra CPU'ya aktarılması sağlanmıştır.
3. Paylaşımlı bellek (shared memory) kullanımı, hesaplamalar esnasında bellek erişimini çokça kullanan fonksiyonlarda önemli derecede performans artışı sağlamıştır.

4. SONUÇ (CONCLUSION)

Son yıllarda, sınıflandırma problemleri için daha verimli çözümlerin getirilmesi üzerine yoğun olarak çalışılmaktadır. Büyük boyutlardaki veri kümelerini yorumlamak ve anlamlandırmak için yoğun hesaplamalı işlemler gerekir. Paralel programlama teknikleri ile zaman alan işlemlerin desteklenmesi hesaplama maliyetini azaltır ve uzmanların işini kolaylaştırır. Bu çalışmada son yıllarda önemi gittikçe artan GPU programlama ile ilgili bir çalışma yapılmıştır. Bu kapsamda büyük ölçekli verilerin sınıflandırılması için genetik algoritma ile optimize edilmiş destek vektör makinesi kullanıldı. Bu yöntemin bir GPU programlama çeşidi olan CUDA Programlama ile hızlandırılması amaçlanmıştır.

CUDA iş parçacığı hiyerarşisini ve bellek hiyerarşisini temel almaktadır. Gerçekleştirilen çalışmadan da görüleceği üzere GA ve DVM gibi yoğun iş yükünün olduğu mimarideki çalışmalar için ve binlerce küçük parçaya bölünebilen sorunlar için CUDA çok uygun gözükmektedir. Çalışmadan elde edilen bir diğer sonuç ise CUDA, en çok kullanılan programlama dillerinden C'nin avantajını iyi kullanarak tüm kodun baştan yazılmasına gerek duymaz. Sadece hesaplama açısından işlem yükü çok olan parçacık için bir CUDA kodu yazabilir ve mevcut koda entegre edilerek gerekli hızlanma sağlanabilmektedir. Uygulamadan çıkarılabilecek bir diğer sonuç ise CUDA ile uygulama geliştirirken, dikkate alınması gereken önemli noktalardan biri, tüm uygulamaların CUDA aygıtı üzerinde iyi ölçeklenemeyeceğidir. Bu noktada sonuçlara göre gelecekteki çalışmalarda, bellek kullanımı ve daha fazla paylaşımlı belleğin kullanılması için kodun optimizasyonuna doğru ilerleyecektir.

KAYNAKLAR (REFERENCES)

[1]. Lo, W. T., Chang, Y. S., Sheu, R. K., Chiu, C. C., & Yuan, S. M. (2014). CUDT: a CUDA based decision tree algorithm. The Scientific World Journal, 2014.

[2]. Sierra-Canto, Xavier, Madera-Ramirez, Francisco, V. Uc-Cetina, Parallel training of a back-propagation neural network using cuda, in: Proceedings of the 2010 Ninth International Conference on Machine Learning and Applications, ICMLA '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 307–312. doi:10.1109/ICMLA.2010.52.

[3]. J. Bhimani, M. Leeser and N. Mi, "Accelerating K-Means clustering with parallel implementations and GPU computing," in High Performance Extreme Computing Conference (HPEC), 2015 IEEE, 2015, pp. 1-6.

[4]. J. Zhang, G. Wu, X. Hu, S. Li and S. Hao, "A parallel K-Means clustering algorithm with MPI," in Parallel Architectures, Algorithms and Programming (PAAP), 2011 Fourth International Symposium on, 2011, pp. 60-64.

[5]. B. Catanzaro, N. Sundaram, and K. Keutzer, \Fast support vector machine training and classification on graphics processors," in Proceedings of the 25th international conference on Machine learning, ICML '08, (New York, NY, USA), pp. 104{111, ACM, 2008.

[6]. L. J. Cao, S. S. Keerthi, C.-J. Ong, J. Q. Zhang, U. Periyathamby, X. J. Fu, and H. P. Lee, \Parallel sequential minimal optimization for the training of support vector machines," Neural Networks, IEEE Transactions on, vol. 17, pp. 1039{1049, July 2006.

[7]. T. He, Z. Dong, K. Meng, H. Wang, Y. Oh, Accelerating multi-layer perceptron based short term demand forecasting using graphics processing units, in: Transmission & Distribution Conference & Exposition: Asia and Pacific, 2009, IEEE, 2009, pp. 1–4.

[8]. Wang, L., Zhu, X., Yang, B., Guo, J., Liu, S., Li, M., & Abraham, A. (2018). Accelerating nearest neighbor partitioning neural network classifier based on CUDA. Engineering Applications of Artificial Intelligence,68, 53-62.

[9]. Krawczyk, Bartosz. "GPU-accelerated extreme learning machines for imbalanced data streams with concept drift." Procedia Computer Science 80 (2016): 1692-1701.

[10]. Singh, Amreek, Kusum Deep, and Pallavi Grover. "A novel approach to accelerate calibration process of a k-nearest neighbours classifier using GPU." Journal of Parallel and Distributed Computing 104 (2017): 114-129.

[11]. Ayşe Arslan, Baha Şen, Musa Peker, Büyük Veri Kümeleri için GPU Tabanlı Paralel Destek Vektör Makinesi, International Conference on Computer Science and Engineering, Tekirdağ, Türkiye, 2016, pp. 519-523

[12]. Basturk, R. Akay, "Performance Analysis of the Coarse-Grained Parallel Model of the Artificial Bee Colony Algorithm", Information Sciences, vol. 253, pp. 34-55, 2013.

[13]. Basturk, R. Akay, "Parallel Implementation of Synchronous Type Artificial Bee Colony Algorithm for Global Optimization", Journal of Optimization Theory and Applications, vol. 155(3), pp. 1095-1104, 2012.

[14]. S. Aslan, H. Badem, T. Ozcan, D. Karaboğa, A. Basturk, "Image Compression with Multi-GPU Accelerated Discrete Haar Wavelet Tramsform", Kahramanmaras Sutcu Imam University Journal of Engineering Sciences, vol. 18, pp.12-16, 2015.

[15]. S. Aslan, H. Badem, T. Ozcan, A. Basturk, D. Karaboga, "CUDA ile Hızlandırılmış İki Boyutlu Ayrık Kosinüs Dönüşümü", 4. Ulusal Yüksek Başarımlı Hesaplama Konferansı, Ankara, Türkiye, 2015.

[16]. S. Aslan, H. Badem, T. Ozcan, A. Basturk, "CUDA Platformunda İki Boyutlu Ayrık Haar Dalgacık Dönüşümü", Akıllı Sistemlerde Yenilikler ve Uygulamaları Sempozyumu (ASYU-2014), pp. 219-222, İzmir, Türkiye, 2014.

[17]. Ruiz-Gonzalez, R.; Gomez-Gil, J.; Gomez-Gil, F.J.; Martínez-Martínez, V. An SVM-Based Classifier for Estimating the State of Various Rotating Components in Agro-Industrial Machinery with a Vibration Signal Acquired from a Single Point on the Machine Chassis. Sensors 2014, 14, 20713-20735.

[18]. Liu M., Wu C., (2003), "Scheduling algorithm based on evolutionary computing in identical parallel machine production line", Robotics and Computer Integrated Manufacturing, 19, 6-7.

[19]. Lessmann, S., Stahlbock, R., and Crone, S. F.: Optimizing hyperparameters of support vector machines by genetic algorithms, In IC-AI, 74–82, 2005

[20]. Pourbasheer, E., Riahi, S., Ganjali, M. R., and Norouzi, P.: Application of genetic algorithm support vector machine (GA-SVM) for prediction of BK-channels activity, Euro. J. Medicinal Chem., 44, 5023–5028, 2009.

[21]. NVIDIA CUDA, https://docs.nvidia.com/cuda/, Erişim Tarihi: 10.12.2017