ADNAN MENDERES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
MATEMATİK ANABİLİM DALI
2014-YL-000...

VEKTÖR NİCEMLEME İÇİN GEOMETRİK BİR ÖĞRENME ALGORİTMASININ TASARIMI VE UYGULAMASI

İclal GÖR

Tez Danışmanı:
Yrd. Doç. Dr. Korhan GÜNEL

AYDIN

ADNAN MENDERES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRLÜĞÜNE
AYDIN

Matematik Anabilim Dalı Yüksek Lisans Programı öğrencisi İclal GÖR tarafından hazırlanan Vektör Nicemleme İçin Geometrik Bir Öğrenme Algoritmasının Tasarımı ve Uygulaması başlıklı tez, 06.08.2014 tarihinde yapılan savunma sonucunda aşağıda isimleri bulunan jüri üyelerince kabul edilmiştir.

	Ünvanı Adı Soyadı	Kurumu	İmzası
Başkan :	Yrd. Doç. Dr. Korhan GÜNEL	ADÜ Fen-Ed. Fak.	
Üye :	Yrd. Doç. Dr. Mine Aylin BAYRAK	KOÜ Fen-Ed. Fak.	
Üye :	Yrd. Doç. Dr. Rıfat AŞLIYAN	ADÜ Fen-Ed. Fak.	

Jüri üyeleri tarafından kabul edilen bu Yüksek Lisans tezi, Enstitü Yönetim Kurulunun ..... sayılı kararıyla .../.../2014 tarihinde onaylanmıştır.

Prof. Dr. Cengiz ÖZARSLAN
Enstitü Müdürü

ADNAN MENDERES ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ MÜDÜRLÜĞÜNE
AYDIN

Bu tezde sunulan tüm bilgi ve sonuçların, bilimsel yöntemlerle yürütülen gerçek deney ve gözlemler çerçevesinde tarafımdan elde edildiğini, çalışmada bana ait olmayan tüm veri, düşünce, sonuç ve bilgilere bilimsel etik kuralların gereği olarak eksiksiz şekilde uygun atıf yaptığımı ve kaynak göstererek belirttiğimi beyan ederim.

06.08.2014

İclal GÖR

ÖZET

VEKTÖR NİCEMLEME İÇİN GEOMETRİK BİR ÖĞRENME ALGORİTMASININ TASARIMI VE UYGULAMASI

İclal GÖR

Yüksek Lisans Tezi, Matematik Anabilim Dalı
Tez Danışmanı: Yrd. Doç. Dr. Korhan GÜNEL
2014, 73 sayfa

Bu çalışmada, makine öğrenmesi alanında sıklıkla kullanılan yöntemlerden biri olan vektör nicemleme yaklaşımındaki serbest parametre sayısı ve referans vektörlerinin hesaplatılmasındaki yoğun iş gücünü azaltarak çözüme daha hızlı yakınsayacak geometrik bir öğrenme algoritması önerilmiştir. Öğrenme algoritmasının temel prensibi, sınıf sınırlarını belirleyen referans vektörlerinin sadece girdi vektörüne değil, paralel olarak dahil olduğu sınıf merkezine yaklaştırılıp uzaklaştırılması esasına dayanır.

Çalışma temel olarak beş bölümden oluşmaktadır. Giriş bölümünde, makine öğrenmesi alanında karşılaşılan sınıflandırma probleminin matematiksel tanımı verilmiş ve sınıflandırma problemlerinin çözümü için literatürde yer alan geometrik yaklaşımlardan bahsedilmiştir. İkinci bölümde bir makine öğrenmesi yaklaşımı olan destekleyici öğrenmeli vektör nicemleme ağlarından ve bu metodun tarihsel gelişiminden söz edilmiştir.

Çalışmanın üçüncü bölümü olan vektör nicemleme için geometrik bir öğrenme algoritması kısmında, yarışmacı öğrenme modelindeki genelleştirilmiş delta öğrenme kuralı kullanımında gerçekleşen problem açıklanmıştır. Bu problemi çözme amacıyla geometrik bir model önerilmiştir ve önerilen yeni metodun çalışma prensibinden bahsedilmiştir. Dördüncü bölümde ise, oluşturulan öğrenme algoritmasının geçerliliğini ispatlamak amacıyla deneysel çalışmalar yapılmış ve önerilen algoritma literatürdeki mevcut algoritmalarla karşılaştırılmıştır. Deneysel çalışmalar sonucunda elde edilen bulgular çalışmanın son bölümünde yorumlanmıştır.

Anahtar Sözcükler
Makine öğrenmesi, Öğrenme algoritması, Vektör nicemleme, Kohonen vektörleri, Geometrik öğrenme yaklaşımı, LVQ

ABSTRACT

A DESIGN AND IMPLEMENTATION OF GEOMETRICAL LEARNING ALGORITHM FOR VECTOR QUANTIZATION

İclal GÖR

M.Sc. Thesis, Department of Mathematics
Supervisor: Asst. Prof. Korhan GÜNEL
2014, 73 pages

In this thesis, the learning vector quantization, one of the frequently used methods in machine learning, is examined in detail. Furthermore, an alternative model with a geometrical approach is proposed to reduce workload and to increase the speed of convergence to the solution of classification problem, by eliminating some arbitrary parameters. The main principle of the proposed learning algorithm is that the boundaries of the classes are determined by moving the reference vectors to or away from not only the sample input vector but also the centroid of the classes using reference hyperspheres.

The thesis is organized into five chapters. The first chapter introduces the classification problem in machine learning from a strictly mathematical viewpoint. Furthermore, some geometrical approaches for solving the classification problems in the literature are mentioned in the same chapter.

Chapter 2 lays out the mathematical foundation of the learning vector quantization, one of the neural network models designed specifically for the classification problem. In the Chapter 3, a fundamental problem encountered, when the generalized learning rule is applied, in some competitive approaches is explained. In order to solve this problem, a geometrical learning approach is presented in the Chapter 3.

In Chapter 4, the proposed method is compared with some variants of learning vector quantization via some experimental studies. The observations obtained with experimental studies are discussed.

Key Words
Machine Learning, Learning algorithm, Vector quantization, Kohonen vectors, Geometrical learning approach, LVQ

ÖNSÖZ

Çalışmamda yardımını, sabrını, anlayışını ve bilgisini esirgemeyen değerli hocam sayın Yard. Doç. Dr. Korhan GÜNEL'e teşekkürü bir borç bilirim.

2210-C Öncelikli Alanlara Yönelik Yurt İçi Yüksek Lisans Burs Programı ile çalışmamı destekleyen Türkiye Bilimsel ve Teknik Araştırma Kurumuna (Tübitak) ve 12006 numaralı Öğretim Görevlisi Yetiştirme Programı (ÖYP) Projesi ile teze destek veren Yükseköğretim Kuruluna (YÖK), sayın Prof. Dr. Gonca GÜNGÖROĞLU ve Yard. Doç. Dr. Rıfat AŞLIYAN başta olmak üzere tezin yazımıyla ilgili yardımcı olan bölüm hocalarıma ve arkadaşlarıma, Öğretim Üyesi sayın Yard. Doç. Dr. Mine Aylin BAYRAK (Kocaeli Üniversitesi) ve Doç. Dr. Emine CAN'a (Kocaeli Üniversitesi) teşekkürlerimi sunarım.

Bugünlere gelmemde büyük pay sahibi olan aileme ve çalışmamın hazırlanmasında beni cesaretlendiren ve manevi desteğini esirgemeyen hayat arkadaşıma teşekkür ederim.

İclal GÖR

İÇİNDEKİLER

SİMGELER DİZİNİ

$\mathbf{w}$	Referans vektörü
$\mathbf{w}_l$	Yerel kazanan referans vektörü
$\mathbf{w}_g$	Global kazanan referans vektörü
$\mathbf{p}$	Girdi vektörü
$\mathbf{m}_A$	A sınıfı ile etiketlenmiş örneklerin merkezi
$\mathbf{c}$	Hiper-kürenin merkezi
λ	Öğrenme oranı
t	İterasyon sayısı
ε	Sabit momentum katsayısı
μ	Sınıflandırıcı fonksiyon
s	Bağıl pencere genişliği
E	Maliyet Fonksiyonu
d_l	Girdi vektörü ile yerel kazanan nöron arasındaki mesafe
d_g	Girdi vektörü ile global kazanan nöron arasındaki mesafe
d_r	r_i ilişki çarpanı olmak üzere, $\mathbf{x}$ girdi vektörü ve $\mathbf{w}$ referans vektörü arasındaki bağıl uzaklık değeri

ŞEKİLLER DİZİNİ

Sayfa No

ÇİZELGELER DİZİNİ

Sayfa No

EKLER DİZİNİ

Sayfa No

KISALTMALAR

GFRLVQ	:	Generalized Functional Relevance Learning Vector Quantization
GLVQ	:	Generalized Learning Vector Quantization
GMLVQ	:	Generalized Matrix Learning Quantization
GRLVQ	:	Generalized Relevance Learning Vector Quantization
LVQ	:	Learning Vector Quantization
LVQRKV	:	Learning Vector Quantization on Rotating Kohonen Vectors
OLVQ	:	Optimized Learning Vector Quantization

1. GİRİŞ

1.1. Makine Öğrenmesi ve Sınıflandırma Problemi

Öğrenme, insanın zamanla kazandığı tecrübelere göre gelişim sürecidir. Bu süreçte insan, elde ettiği bilgileri kullanarak yeni bilgiler öğrenir. Bilgisayarlı öğrenme için bu süreç benzer şekilde işlemektedir. Kısaca makine öğrenmesi, bilgisayara insan beynine benzer şekilde öğrenme yeteneğinin kazandırılmasıdır.

Bilgisayarda algoritma kullanılarak yapılan işler, belirli sırada ve hata payı olmadan gerçekleşir. Girdiyi çıktıya çevirme amacıyla oluşturulan bu komutlar dizisi olmadan, bilgisayarın örnek verilerden ya da edindiği deneyimlerden yararlanarak karar verme sürecinin gerçekleştiği işler de vardır. Örneğin, elektronik posta ayıklanması işleminde girdi elektronik posta ve çıktı ise evet ya da hayır işaretidir. Dolayısıyla girdiye bağlı çıktı tespitinde bilgisayarın karar verme süreci söz konusudur. Aynı zamanda bilgisayarlar, insanın yapabileceği hatalar gibi gereksiz elektronik posta ayıklama sürecinde yanlış karar verebilir. Kısaca yapay öğrenme, bilgisayarın örnek verilerden ya da deneyimlerden faydalanarak karar verebilmesidir [1].

Makine öğreniminin literatürde benzer anlam içeren farklı tanımları mevcuttur. Arthur Samuel (1959), makine öğrenmesi tanımını 'Bilgisayarın programlama olmadan öğrenme yeteneği' olarak tanımlamıştır [2].

Tom M. Mitchell çok daha genel bir tanım vererek 'Bilgisayarlar, E kümesi ile gösterilen deneyimleri kullanarak bir T kümesindeki görevleri P kümesi ile belirtilen performansları gerçekleştirerek öğrenirler.' şeklinde bilgisayar öğrenimini ifade etmiştir [3].

Alan Turing (1950), Mind dergisinde yayınlanan 'Computing Machinery and Intelligence' adlı makalesinde 'Bilgisayarlar düşünebilir mi? (Can machines

think?)' sorusuyla makine öğrenimine dikkat çekmiştir. Bilgisayar öğreniminin insan beynindeki öğrenime benzer şekilde gerçekleştiğinin önemini ise 'Bizim düşündüğümüz gibi bilgisayarlar düşünebilir mi? (Can machines do what we (as thinking entities) can do?)' sorusuyla vurgulamıştır [4].

Tıpkı insanların farklı öğrenme stilleri olabildiği gibi, makine öğreniniminin de destekleyici (gözetimli), yarı destekleyici (yarı gözetimli) ve destekleyici olmayan (gözetimsiz) olmak üzere üç farklı öğrenme stratejisi mevcuttur. Destekleyici öğrenmede, Tanım 1.1 ile verildiği üzere, girdi değeri ile gözetmen tarafından tespit edilen beklenen çıktı değerinin eşleme yapılması söz konusudur. Bu yaklaşımda gözetmen girdi seti için beklenen çıktı değerini sisteme göstermez. Bunun yerine sistemin ürettiği çıktıların doğru veya yanlış olduğunu söyler. Buna göre sistem de öğrenmeye devam eder.

Tanım 1.1 $i = 1, 2, ..., n$ için $\mathbf{x_i}$ öznitelik vektörü ve y_i etiket olmak üzere, $D = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), ..., (\mathbf{x_n}, y_n)\}$ eğitim kümesi verilmiş olsun. Buna göre sistemin ürettiği çıktıların doğru ya da yanlış olduğunun gözetmen tarafından tespit edilme sürecinin gerçekleştiği öğrenme stratejisine destekleyici (gözetimli) öğrenme denir.

Diğer bir öğrenme stratejisi Tanım 1.2 ile verilen yarı destekleyici öğrenmede, az sayıda etiketlenmiş ve çok sayıda etiketlenmemiş veriler vardır. Bu öğrenme stilinde hangi girdi değerinin hangi sınıfa ait olduğu bilinmemektedir. Öncelikle ağ eğitilir ve girdi değerleri hangi etikete ait olduğu tespit edilir. Yani her bir örnek için hem girdiler hemde oluşturulması beklenen çıktılar sisteme gösterilmektedir. Bu durumda sistem gözetmenin belirlediği çıktılara göre girdi değerlerini haritalar.

Tanım 1.2 X girdi seti ve Y çıktı seti olsun. Buna göre sisteme girdi ve çıktılar gösterilir ve sistem gözetmenin gösterdiği çıktılara göre haritalama yapması beklenir. Bu öğrenme stratejisine yarı destekleyici (yarı gözetimli) öğrenme denir.

Tanım 1.3 ile verilen destekleyici olmayan öğrenme modelinde ise, sadece girdi değerlerini içeren eğitim kümesi verilir. Bu öğrenme modelinde gözetmen bulunmaz. Sistemin kendi kendisine öğrenmesi beklenmektedir. Öğrenme sonucunda çıktıların anlamı kullanıcı tarafından belirlenir. Bu yaklaşım daha çok sınıflandırma problemlerinde kullanılır.

Tanım 1.3 $D = \{\mathbf{x_n} \mid n = 1,2,...,N\}$ eğitim kümesi verilsin. Sistem bu eğitim kümesine göre öğrenmeyi gerçekleştirir ve kullanıcı tarafından çıktıların karar verildiği öğrenme stratejisine destekleyici olmayan (gözetimsiz) öğrenme denir.

Bu çalışmada geliştirilen vektör nicemleme yöntemi ise, bir destekleyici (gözetimli) öğrenme metodudur [5].

Örüntü tanıma, kümeleme, karar verme gibi problemlerin çözümünde farklı öğrenme stratejilerine sahip makine öğrenmesi yaklaşımları kullanılmaktadır. Bu uygulama alanları dışında sınıflandırma problemlerinin çözümü için de makine öğrenmesi iyi bir yaklaşımdır. Sınıflandırma problemi aşağıdaki gibi ifade edilebilir:

Farklı sınıflara ait nesneler verilmiş olsun. Yeni karşılaşılan bir nesneyi bu sınıflardan birine atama problemi sınıflandırma problemi olarak adlandırılır. İki farklı sınıf bulunduğunu varsayalım. Bu durumda sınıflandırma problemi için makine öğrenmesi Tanım 1.4 ile verildiği gibi ifade edilebilir.

Tanım 1.4 X, $\mathbf{x_i}$ gözlemlerinin alındığı boştan farklı bir küme ve y_i etiket (çıktı) olmak üzere; $(\mathbf{x_1}, y_1), ..., (\mathbf{x_m}, y_m) \in X \times \{\pm 1\}$ sınıflandırma problemi için makine öğrenmesidir.

Bu problemde gözlemler $+1$ ve -1 olarak etiketlenmiş iki sınıfa aittir. Bu tip sınıflandırmalara **ikili tip sınıflandırma** denir. Sınıflandırma problemlerinde, öğrenme gerçekleştiği zaman daha önceden bilinmeyen verileri genelleştirebilmek istenmektedir. Örüntü tanıma probleminde bunun anlamı yeni bir $\mathbf{x} \in X$ verildiğinde buna karşılık gelen $y \in \{\pm 1\}$ etiketini tahmin etmektir. Bunun için X ve $\{\pm 1\}$ üzerinde Tanım 1.5 ile tanımlanan benzerlik ölçütlerine ihtiyaç vardır.

Tanım 1.5 X evrensel küme ve $S \subset X$ örnek küme olmak üzere; $f : X \to \{\pm 1\}$ eğitim fonksiyonunu tanımlayalım. $D=\{(\mathbf{x},y) | \ \mathbf{x} \in S \ ve \ y = f(\mathbf{x})\}$ eğitim kümesi için $\widetilde{f}(\mathbf{x})$ fonksiyonunu tanımlama işlemi benzerlik ölçütüdür.

Benzerlik ölçütü bulma işlemi, $\mathbf{x} \in S$ için $f(\mathbf{x})$ biliniyorken, $\mathbf{x} \in X$ için $\widetilde{f}(\mathbf{x})$ fonksiyonunu belirleme işlemidir. Yani, sınıflandırma problemlerinde benzerlik ölçütü kullanılarak çözüm yapılmaktadır. Buna göre, makine öğrenmesi Tanım 1.6 ile açıklandığı gibi ifade edilebilir.

Tanım 1.6 X evrensel küme ve $S \subset X$ örnek küme olmak üzere; $\forall \mathbf{x} \in S$ için $\widetilde{f}(\mathbf{x})$ biliniyorken $\forall x \in X$ için $f(\mathbf{x})$ fonksiyonunu bulma işlemine makine öğrenmesi denir.

Makine öğrenmesindeki problemlerden biri olan sınıflandırma probleminin kullanıldığı birçok alan mevcuttur. Güvenlik kontrolünde kullanılan sınıflandırma problemlerinden bazıları, parmak izi tespiti, göz retinasını tarayarak yüz tanıma, farklı profillerden çekilmiş fotoğrafı kullanarak elde edilen kişi tanıma, imzaların taklit edilip edilmediğini öğrenme, güvenlik kamerasında olağandışı bir olayın olup olmadığını kontrol etmedir. Diğer bir uygulama alanı ise gereksiz e-posta tespitidir. Makalede çalıntı olup olmamasını tespit etme yine bir sınıflandırma problemidir. Sağlık alanında kullanılan sınıflandırma problemleri ise, tomografi görüntüsünde tümör tespiti, kanser tespiti ve meme kanseri tespiti gibi uygulama alanlarıdır.

Sınıflandırma problemleri için uygulama alanları, yapılan yeni çalışmalarla birlikte günümüzde gittikçe geniş alana yayılmaktadır.

1.2. Sınıflandırma Problemi için Geometrik Yaklaşımlar

İstatiksel öğrenme teorisinde, sınıflandırma problemlerinin çözümü için literatürde bir çok geometrik yaklaşım mevcuttur [6]. Bu geometrik yaklaşımlardan biri, CHILS (Convex Hull Inductive Learning System) adında sınıflandırma problemi çözümü için konvekslerin kullanıldığı öğrenme algoritmasıdır [7]. Diğer bir öğrenme algoritması, lineer ayrılamayan fonksiyonu lineer ayrılabilen fonksiyona dönüştüren ETL (Expand and Truncate Learning) yaklaşımıdır [8]. Bu yaklaşım, tamsayı değerlerine sahip ağırlık ve eşik değerlerini içerir ve destekleyici öğrenmeli vektör nicemleme ağlarının öğrenme kuralı olan genelleştirilmiş delta öğrenme kuralına göre sonuca daha hızlı yakınsar. Cabrelli, Molter, ve Shonkwiler (2000) tarafından, iki katmanlı ağlar kullanılarak sınıflandırma işlemi gerçekleştirebilen CoRD (Convex Recursive Deletion Regions) modeli oluşturulmuştur [9].

Boolean sinir ağları için girdi uzayındaki örnekleri üç eş merkezli referans kürelerin kesişimini kullanarak sınıflandıran FCLA (Fast Covering Learning Algorithm) metodu önerilmiştir [10]. Shoujue ve Jiangliang (2005), tasarı geometri ile, yani örneklerin yüksek boyutlu yapılarının düşük boyuta yansıtma aracılığıyla sınıflandırma gerçekleştiren bir öğrenme algoritması oluşturmuştur [11].

Bayro-Corrochano ve Anana-Daniel (2005), Clifford geometrik cebir kullanan reel ve kompleks değerli destek vektör makinesi (SVM) yönteminin genelleştirilmiş hali olan Cliffer Destek Vektör Makinesi (CSVM - Cliffer Support Vector Machine) yöntemini önermiştir. Önerilen yaklaşım çok sınıflı sınıflandırma problemleri için etkin bir yöntemdir [12].

Zhang, Chan ve Lee (2005), metin sınıflandırma problemlerinin çözümü için multinomial manifold kullanmıştır. Önerilen metotta metin sınıflandırma için öklid geometrisi kullanılmıştır [13]. Delogu, Fanni ve Montisci (2008), doğrusal

programlama kullanarak belirlenen minimal hacimli polihedronlar ile desen sınıflandırma probleminin çözümüne uygun öğrenme algoritması tanımlamıştır [14].

Liu ve dig. (2009), örüntü tanıma ve makine öğrenimi konusunda sıklıkla kullanılan optimizasyon problemleri için geometrik yaklaşımlardan faydalanmış ve SCH (scaled convex hull) kullanarak ayrılmayan sınıflandırma problemlerinin çözümünde teorik bir yaklaşım önermişlerdir [15]. Wang ve dig. (2013), Convex Hull tepe noktası seçimine dayalı çevrimiçi destek vektör makinesi metodunu önermiştir. Bu metoda göre, seçilen girdi örnekleri Convex Hull için tepe noktası olarak belirlenir [16]. Günümüzde sınıflandırma problemleri konusunda geometrik yaklaşım kullanan çalışmalar yapılmaya devam edilmektedir.

Sınıflandırma problemlerinin çözümü için kullanılan yaklaşımlardan biri olan vektör nicemleme ağları ve gelişimi bir sonraki bölümde incelenecektir.

2. DESTEKLEYİCİ ÖĞRENMELİ VEKTÖR NİCEMLEME AĞLARI (LVQ)

2.1. LVQ Ağları

Vektör nicemleme algoritması (LVQ), 1989 yılında Tuevo Kohonen tarafından geliştirilen ve makine öğrenmesi alanında kullanılan bir yapay sinir ağı modelidir. Ağ topolojisi temel olarak sınıflandırma problemlerinin çözümüne uygun olarak geliştirilmiştir. Bu yaklaşıma göre, örneklem uzayındaki her bir sınıf referans vektörleri tarafından temsil edilir. LVQ ağı topolojisi, girdi katmanı, Kohonen katmanı ve çıktı katmanı olmak üzere üç katmandan oluşur ve Kohonen katmanında yer alan nöronlar (referans vektörleri) birbirleriyle yarışır. Yaklaşım 'kazanan her şeyi alır' prensibine göre çalışır. Kazanan nöron, girdi vektörüne en yakın olan nörondur.

Vektör nicemleme ağlarında eğitim süreçlerinin ilk adımı, Kohonen katmanındaki referans vektörlerinin ilk değerlerinin veya ağırlıklarının verilmesi işlemidir. Ağırlıklar başlangıçta ya sabit bir değer olarak seçilir veya belli bir $a,b \in \mathbb{R}$ için, [a,b] aralığında rastgele atanır. Bu işlemden sonra, ağa herhangi bir sınıfa ait örnek gösterildiğinde kazanan nöronun belirlenebilmesi için Kohonen katmanında yer alan nöronlar ile girdi vektörünün arasındaki uzaklıklar hesaplanır. Bu uzaklık genellikle Öklid mesafesi kullanılarak hesaplanır. Kazanan nöronun tespitinden sonra ağırlığının güncellenmesi işlemi, n boyutlu uzayda Kohonen vektörlerinin sayısı M olmak üzere, $k = \arg\min_{1\leq i\leq M}\left\{\sum_{j=1}^{n}(x_j - w_{ij})^2\right\}$ koşulu altında, (2.1.1) eşitliği kullanılarak yapılmaktadır.

$$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) - \lambda(t)(\mathbf{x} - \mathbf{w}_k(t)) \tag{2.1.1}$$

(2.1.1) eşitliğinde, $\mathbf{w}_k$ kazanan vektör olmak üzere, t iterasyon sayısı ve $0 < \lambda < 1$ için serbest parametre olan λ öğrenme katsayısıdır.

2.2. LVQ Ağlarının Tarihçesi

Kohonen tarafından 1989 yılında tanıtılan ilk vektör nicemleme ağı metodu LVQ1 olarak bilinir ve (2.1.1) eşitliğindeki genelleştirilmiş delta öğrenme kuralını kullanır. Daha sonra yapılan çalışmalarda bu model baz alınarak yöntem sürekli geliştirilmiştir. Bu öğrenme algoritmalarından ilki olan LVQ2 algoritması yine Kohonen (1990) tarafından geliştirilmiştir [17, 18]. Bu algoritmada, Kohonen vektörü olarak yerel ve global nöron belirlenmiştir. Yerel kazanan vektör $\mathbf{w}_l$, girdi vektörü $\mathbf{x}$'e en yakın ve $\mathbf{x}$ ile aynı sınıftan olan vektördür. Global kazanan vektör $\mathbf{w}_g$ ise, tüm Kohonen vektörler arasından girdi vektörüne en yakın vektör olarak bilinir. Bu yaklaşıma göre, s keyfi seçilen pencere genişliği olmak üzere, $\min\left(\frac{d_l}{d_g}, \frac{d_g}{d_l}\right) > s$ ve $d_l = |\mathbf{x} - \mathbf{w}_l|$ ve $d_g = |\mathbf{x} - \mathbf{w}_g|$ şartları altında (2.2.2) denklem sistemindeki ilk eşitlik kullanılarak yerel kazanan vektör $\mathbf{w}_l$, girdi vektörü $\mathbf{x}$'e yaklaştırılır. Bu öğrenme metodunda temel fikir, girdi vektörü $\mathbf{x}$'in $\mathbf{w}_l$ ve $\mathbf{w}_g$ arasındaki orta düzlem olarak tanımlı bağıl pencere içine düşmesidir. Kohonen (1996), bağıl pencere genişliği s'nin 0.2 ve 0.3 arasında değişmesi gerektiğini sezgisel olarak belirtmiştir. Ayrıca, (2.2.2) denklem sistemindeki ikinci eşitlik kullanılarak global kazanan $\mathbf{w}_g$ girdi vektörü $\mathbf{x}$'ten uzaklaştırılır. Bu metot LVQ2 metoduna benzer bir metottur ve LVQ2.1 olarak adlandırılır. Diğer yandan bu yaklaşımda kullanılan süreç yakınsamayı garanti altına almaz.

$$\left.\begin{array}{rcl} \mathbf{w}_l(t+1) & = & \mathbf{w}_l(t) - \lambda(t)(\mathbf{x} - \mathbf{w}_l(t)) \\ \mathbf{w}_g(t+1) & = & \mathbf{w}_g(t) + \lambda(t)(\mathbf{x} - \mathbf{w}_g(t)) \end{array}\right\} \tag{2.2.2}$$

(2.2.2) denklem sisteminde, $m < M$ için girdi vektörü $\mathbf{x}$ ile aynı sınıfta olan, i_1 ve i_m arasında indekslenmiş Kohonen nöronları içinden yerel kazanan nöron $l = \arg\min_{i_1 \le i \le i_m} \{\|\mathbf{x} - \mathbf{w}_i\|_2\}$ koşulu ile belirlenir. Global kazanan nöronun indisi ise, $j \notin \{i_1, \ldots, i_m\}$ için $g = \arg\min_{1 \le j \le M} \{\|\mathbf{x} - \mathbf{w}_j\|_2\}$ eşitliği ile elde edilir.

LVQ metodunun diğer bir modeli ise, LVQ2.1 metoduyla $g = \arg\min_{1 \le j \le n} \{\|\mathbf{x} - \mathbf{w}_j\|_2\}$ koşulu haricinde aynı sürece sahip LVQ3 öğrenme

algoritmasıdır [17]. Bu algoritmada amaç, $\mathbf{w_g}$ ile $\mathbf{w_l}$ vektörlerinin aynı sınıf içine düşebilmesinin sağlanmasıdır. Bu amaç doğrultusunda, (2.2.3) eşitliği kullanılarak, $\varepsilon \in (0,1)$ sabit momentum katsayısı olmak üzere, ek bir ağırlık güncellemesi uygulanır. ε değeri $\mathbf{w}_g$ ile $\mathbf{w}_l$ arasındaki sınırı tespit etmede kullanılacak pencere genişliğini etkileyen bir serbest parametredir.

$$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) + \varepsilon\lambda(t)(\mathbf{x}(t) - \mathbf{w}_k(t)) \tag{2.2.3}$$

Diğer yandan LVQ2.1 için karşılaşılan problem LVQ3 algoritması için de geçerlidir [19].

Sato, Yamada (1995) tarafından önerilen GLVQ (Generalized Learning Vector Quantization) algoritmasında, tanımlanan bir $E = \frac{1}{2}\sum_{k=1}^{N} f(\mu(\mathbf{x}_k))$ maliyet fonksiyonu yardımıyla pencere genişliği belirlenir [19]. Maliyet fonksiyonunu minimize etmek amacıyla (2.2.4) eşitliğinde görülen $\mu(\mathbf{x})$ sınıflandırıcı fonksiyonu kullanılmaktadır.

$$\mu(\mathbf{x}) = \frac{d_l - d_g}{d_l + d_g}, \tag{2.2.4}$$

(2.2.4) eşitliğinde, d_l, girdi vektörü $\mathbf{x}$ ile yerel kazanan nöron $\mathbf{w}_l$ arasındaki mesafe, d_g ise girdi vektörü $\mathbf{x}$ ile global kazanan nöron $\mathbf{w}_g$ arasındaki mesafedir.

Maliyet fonksiyonunu minimize ederek, $\mathbf{w}_l$ ve $\mathbf{w}_g$ kazanan vektörlerinin ağırlıklarının güncellenmesi işlemi (2.2.5) eşitliği kullanılarak gerçekleştirilir ve bu işlem veri setindeki örneklerin toplam sayısı N olmak üzere, adım adım indirgeme yöntemine (steepest descent method) dayalıdır.

$$\left.\begin{array}{rcl} \mathbf{w}_l & = & \mathbf{w}_l + \lambda \frac{\partial f}{\mu} \frac{d_g}{(d_l+d_g)^2}(\mathbf{x} - \mathbf{w}_l) \\ \mathbf{w}_g & = & \mathbf{w}_g - \lambda \frac{\partial f}{\mu} \frac{d_l}{(d_l+d_g)^2}(\mathbf{x} - \mathbf{w}_g) \end{array}\right\} \tag{2.2.5}$$

(2.2.5) denklem sisteminde verilen d_l ve d_g değerleri, $d_l = ||\mathbf{x} - \mathbf{w}_l||^2$ ve $d_g = ||\mathbf{x} - \mathbf{w}_g||^2$ olarak hesaplanır.

GLVQ metodunda, yakınsama maliyet fonsiyonunun tanımına bağlıdır ve sigmoid fonksiyonu, $f(\mu,t) = 1/(1+\exp(-\mu t))$ olarak seçilebilir. OLVQ1 ve OLVQ2

olarak bilinen ve sırasıyla LVQ1 ve LVQ3 metotlarının iyileştirilmiş versiyonu olan yaklaşımlarında uygulanan süreç LVQ1 ve LVQ3 metotlarıyla benzerdir; ancak her bir Kohonen vektörünün kendilerine ait bir öğrenme oranı mevcuttur. Öğrenme oranının değerleri $1 \leq i \leq M$ için (2.2.6) eşitliğindeki gibi bir özyineleme fonksiyonu aracılığıyla belirlenir [19].

$$\lambda_i(t) = \begin{cases} \dfrac{\lambda_i(t-1)}{1+\lambda_i(t-1)}, & \mathbf{w}_i \text{ ve } \mathbf{x} \text{ aynı sınıfta ise,} \\ \dfrac{\lambda_i(t-1)}{1-\lambda_i(t-1)}, & \text{Aksi halde} \end{cases} \tag{2.2.6}$$

Kohonen vd. (1996), OLVQ1 ve OLVQ3 metotlarının LVQ1 ve LVQ3'e göre daha hızlı yakınsama sağladığını; ancak bu yaklaşımlarda kullanılan tekniğin, bazı durumlarda çözümden uzaklaşıldığı için, LVQ2 ve LVQ2.1 metotlarına uygulanamaz olduğunu iddia etmiştir [20].

Bojer vd. (2001), önerdikleri Relevance Learning Vector Quantization (RLVQ) adlı vektör nicemleme yaklaşımında, öğrenme katsayısını bir vektör olarak tanımlamışlardır [21]. Bu yaklaşımda n girdi vektörünün boyutu olacak şekilde, öğrenme katsayısı λ_i vektörü bileşenlerinin başlangıç değeri $\dfrac{1}{n}$ olarak belirlenir. Öğrenme katsayısı vektörünün güncellenmesi işlemi ise, $\alpha \in (0,1)$ girdi ağırlıkları için öğrenme katsayısı olmak üzere, (2.2.7) ve (2.2.8) eşitliği ile gerçekleştirilir.

$$\lambda_l = \begin{cases} \max\{\lambda_l - \alpha|x_l(i) - w_l(j)|, 0\}, & y^i = c^j \text{ ise} \\ \lambda_l + \alpha|x_l(i) - w_l(j)|, & \text{Aksi halde} \end{cases} \tag{2.2.7}$$

$$\lambda_l = \frac{\lambda_l}{|\lambda_l|} \tag{2.2.8}$$

RLVQ öğrenme algoritmasının üç farklı çeşidi vardır. Bunlardan ilki RLVQi olarak adlandırılır ve öncelikle $\mathbf{w}$ ağırlık vektörleri, Kohonen tarafından önerilen (2.2.9) eşitliği ile ve λ öğrenme oranı ise (2.2.7) eşitliği ile güncellenir. RLVQii adıyla bilinen diğer bir öğrenme algoritmasında ise, $\mathbf{w}$ ve λ, (2.2.9) eşitliği kullanılarak güncelleme işlemi gerçekleştirilir. RLVQiii metodunda ise önce (2.2.9) eşitliği ile λ öğrenme katsayısının güncelleme işlemi ve sonrasında (2.2.7) eşitliği ile $\mathbf{w}$

ağırlık vektörlerinin güncelleme işlemi gerçekleştirilir.

$$w^j = \begin{cases} w^j + \varepsilon(x^j - w^j), & y^i = c^j \text{ ise} \\ w^j - \varepsilon(x^j - w^j), & \text{Aksi halde} \end{cases} \tag{2.2.9}$$

Vektör nicemleme metodunun diğer bir çeşidi olan GRLVQ (Generalized Relevance Learning Vector Quantization) metodu, GLVQ ve RLVQ algoritmalarının iyileştirilmesiyle oluşturulmuş yöntemdir [22]. Bu yaklaşıma göre, **x** ve **w** arasındaki mesafe (2.2.10) eşitliğinde verilen metrik kullanılarak hesaplanır.

$$d_r(\mathbf{x}, \mathbf{w}) = \sum_i r_i (x_i - w_i)^2 \tag{2.2.10}$$

(2.2.10) eşitliğinde görülen r_i değeri, başlangıçta $r_i < 0$ ve $\sum_i r_i = 1$ olmak üzere, ilişki çarpanıdır. İlişki çarpanının kullanılması girdi boyutlarının ölçeklendirilmesini, yani gerçek bir örneği temsil eden öznitelik vektörü olarak tanımlı girdi vektörünün hangi bileşeninin sınıflandırmada daha belirleyici olduğunun tespit edilmesini sağlar. GLVQ metodunun bağıl öğrenme sürecinde,

$$\frac{\partial_S E}{\partial d_l} = \frac{2 d_g . f'(\mu(\mathbf{w}))}{(d_l + d_g)^2}$$

ve

$$\frac{\partial_S E}{\partial d_g} = -\frac{2 d_l . f'(\mu(\mathbf{w}))}{(d_l + d_g)^2}$$

olmak üzere, (2.2.11) eşitlikleri kullanılarak maliyet fonksiyonu minimize edilir.

$$\frac{\partial_S E}{\partial \mathbf{w}_l} = \frac{\partial_S E}{\partial d_l} \frac{\partial d_l}{\partial \mathbf{w}_l}, \frac{\partial_S E}{\partial \mathbf{w}_g} = \frac{\partial_S E}{\partial d_g} \frac{\partial d_g}{\partial \mathbf{w}_g} \tag{2.2.11}$$

Bağıl ağırlıklar ise, ∂_S stokastik gradyan düşüm,

$$\frac{\partial_S E}{\partial \mathbf{w}_l} = \frac{\partial_S E}{\partial d_l} \frac{\partial d_l}{\partial \mathbf{w}_l}$$

ve

$$\frac{\partial_S E}{\partial \mathbf{w}_g} = \frac{\partial_S E}{\partial d_g} \frac{\partial d_g}{\partial \mathbf{w}_g}$$

olmak üzere $0 < \varepsilon_r < 1$ ve

$$\frac{\partial E_S}{\partial r_i} = \frac{2 d_g . f'(\mu(\mathbf{w}))}{(d_l + d_g)^2} \frac{\partial d_l}{\partial r_i} - \frac{2 d_l . f'(\mu(\mathbf{w}))}{(d_l + d_g)^2} \frac{\partial d_g}{\partial r_i}$$

için (2.2.12) eşitliği aracılığıyla güncellenir.

$$r_i = r_i - \varepsilon_r \frac{\partial_S E}{\partial r_i} \tag{2.2.12}$$

Seo, Obermayer (2003), maliyet fonksiyonu kullanarak, gradyan düşüm uygulayan ve Esnek Vektör Nicemleme Ağı (Soft LVQ) olarak adlandırılan bir öğrenme kuralı geliştirmiştir [23].

GRLVQ metodunun geliştirilmiş versiyonu olan GMLVQ (Generalized Matrix Learning Quantization) algoritmasında, Λ, ikinci dereceden, pozitif ve yarı belirli matris olmak üzere, (2.2.13) eşitliğinde verilen parametrik ikinci derece formu kullanılır [24].

$$d_\Lambda(\mathbf{x}, \mathbf{w}) = (\mathbf{x} - \mathbf{w})^T \Lambda (\mathbf{x} - \mathbf{w}). \tag{2.2.13}$$

Kaestner vd. (2012), vektör nicemleme metodundaki ilişki çarpanlarını güncellemede, bağımsız parametre sayısını azaltmak için GFRLVQ (Generalized Functional Relevance Learning Vector Quantization) yaklaşımını önermiştir [25]. Bu yaklaşımda (2.2.14) eşitliğinde verilen ilgi faktörü olan $r_i = r(t_i)$, Gauss veya Lorentz tipli fonksiyonların lineer birleşimi olarak tanımlanır.

$$r(t_i) = \sum_j \beta_j \kappa_j(\omega_j, t_j) \tag{2.2.14}$$

(2.2.14) eşitliğinde yer alan κ_j, $\omega_j = (\omega_{j,1}, \omega_{j,2}, \ldots, \omega_{j,p})^T$ gibi birkaç parametreye sahip olan temel bir fonksiyondur. Ayrıca $\beta_j > 0$ ve $\sum_j \beta_j = 1$ koşulları sağlanır.

Biehl, Ghosh ve Hammer (2007), çalışmalarında vektör nicemleme ağlarını teorik olarak analiz etmiş ve nasıl geliştirilebileceğine dair ipuçları vermiştir [26]. Literatürde LVQ ağlarıyla ilgili teorik olarak çalışılmaya devam edilmektedir.

Bu tezde bir sonraki bölümde sunulacak olan öğrenme algoritması, pencere genişliği kavramını ortadan kaldırmış, momentum katsayısı, bağıl öğrenme oranları gibi sezgisel olarak seçilen parametrelere olan bağımlılığı azaltmış ve sınıf sınırlarını belirlemede gereksiz iş yükünden kurtararak çözüme daha hızlı yakınsama sağlamıştır.

3. VEKTÖR NİCEMLEME İÇİN GEOMETRİK ÖĞRENME ALGORİTMASI

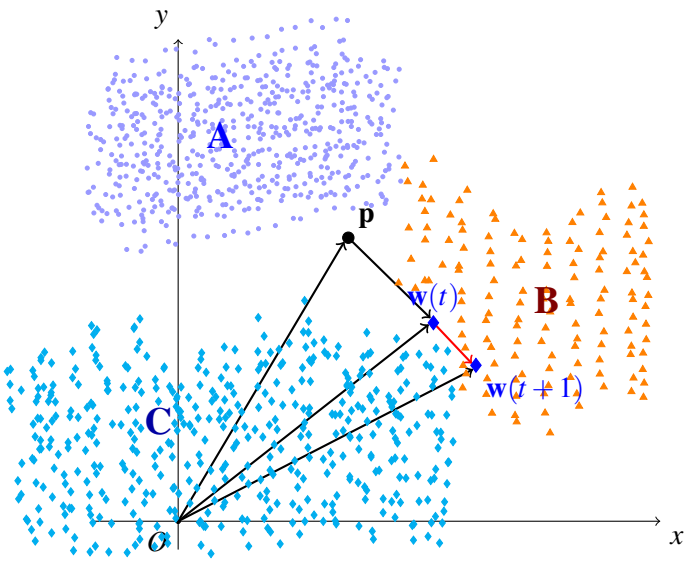
3.1. LVQRKV Metodu

Kohonen öğrenme kuralına göre, referans vektörü $\mathbf{w}_i$ ile girdi vektörü $\mathbf{p}$ arasındaki mesafenin öğrenme oranına göre arttırılması ya da azaltılması gerekir. Genelleştirilmiş delta kuralı ile öğrenmede, bazı durumlarda referans vektörünün girdi vektöründen uzaklaşırken, temsil etmesi gereken sınıftan da uzaklaştığı gözlemlenmiştir. Şekil 3.1'de görüldüğü üzere, $\mathbf{p}$ vektörü A sınıfına ait girdi vektörü ve $\mathbf{w}$ ise C sınıfının bir referans vektörü olsun. Eğer $\mathbf{w}$ global kazanan ise, genelleştirilmiş delta öğrenme kuralına göre $\mathbf{p}$ vektöründen uzaklaşması gerekir. Fakat $\mathbf{p}$ vektöründen uzaklaşırken, ait olduğu C sınıfından da uzaklaşmaktadır ve B sınıfının sınır bölgesine doğru yaklaşmaktadır. Karşılaşılan bu problemin üstesinden gelmek için, yeni bir öğrenme kuralı oluşturulmuştur.

Önerilen öğrenme algoritmasının temel fikri, kazanan nöronun girdi vektörüne yakınlaştırılmasına veya girdi vektöründen uzaklaştırılmasına paralel olarak, daima temsil ettiği sınıf merkezine de yakınlaştırılmasıdır. Bu fikrin kolayca gerçekleştirilebilmesi için küresel koordinat sisteminde, $\mathbf{w}$ referans vektörü oluşturulan bir hiper-küre üzerinde döndürülür. n boyutlu öklidyen uzayda, $(x_1, x_2, \ldots, x_n)$ noktaları verilmiş olsun. S_{n-1} hiper-küre denklemi, $\mathbf{x}$ girdi vektörü, $\mathbf{c}$ merkez ve r ise S_{n-1} hiper-kürenin yarıçapı olmak üzere, (3.1.1) eşitliğinde verilmiştir.

$$\sum_{i=1}^{n} (x_i - c_i)^2 = r^2 \tag{3.1.1}$$

Bu yaklaşım tarafımızdan LVQRKV (Learning Vector Quantization with Rotating Kohonen Vectors) olarak adlandırılmıştır. LVQRKV metodu için, ilk olarak iki farklı hiper-küre oluşturulmuştur. $\mathbf{w}$ kazanan vektör olmak üzere, hiper-kürelerden birinin merkezi girdi vektörü $\mathbf{p}$ ve yarıçapı ise $|\mathbf{pw}|$ olsun. Diğer hiper-küre



Şekil 3.1. LVQ metodunda genelleştirilmiş delta öğrenme kuralı kullanıldığında karşılaşılan problem

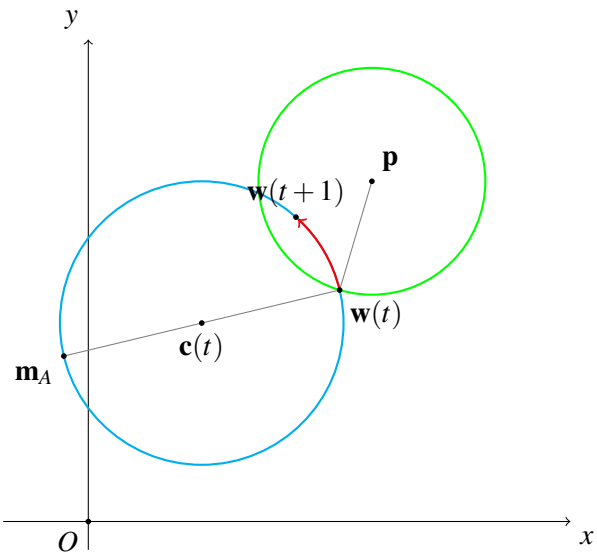
ise, referans vektör **w** ile A sınıfının merkezi arasındaki mesafenin orta noktasını merkez ve $|\mathbf{wm}_A|$ çap kabul eden hiper-küre olsun.

Anlaşılır olması için örnek olarak, Şekil 3.2'de görüldüğü üzere, iki boyutlu uzayda $\{A, B\}$ gibi iki farklı sınıfın verildiğini varsayalım. Eğer **w** vektörü yerel kazanan ise, hem girdi vektörü olan **p** vektörüne hem de A sınıfının merkezine yaklaşmalıdır. Bu amaçla Şekil 3.2 ve Şekil 3.3'te görüldüğü gibi, **w** vektörü için, merkezi **c** ve yarıçapı $|\mathbf{cw}|$ olan hiper-küre üzerinde döndürme işlemi yapılmaktadır. Şekil 3.2'de, dönme sonucunda oluşturulan hiper-küreler için

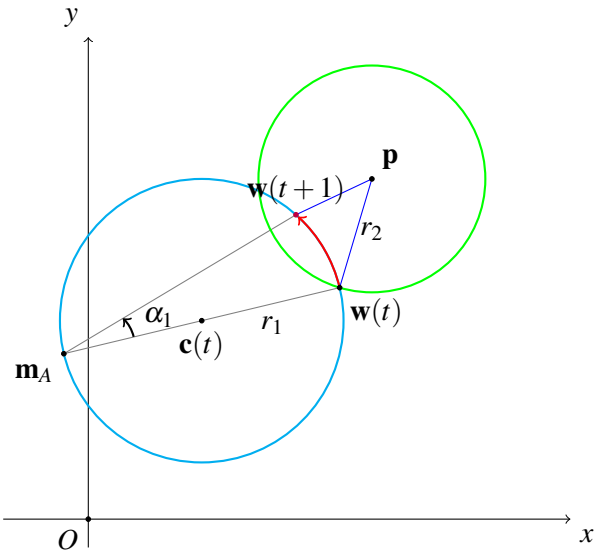
$$\begin{aligned} \sum_{i=1}^{n}(x_i - c_i)^2 &= |\mathbf{cw}|^2, \\ \sum_{i=1}^{n}(x_i - p_i)^2 &= |\mathbf{pw}|^2 \end{aligned}$$

eşitlikleri geçerlidir. Şekil 3.3'te görüldüğü üzere, $|\mathbf{m}_A\mathbf{w}(t+1)| < |\mathbf{m}_A\mathbf{w}(t)|$ ve $|\mathbf{pw}(t+1)| < |\mathbf{pw}|$ olduğundan dönme sonucunda öğrenme garanti altına alınır.

Hiper-küre üzerinde döndürme işlemi için öncelikle referans hiper-küreler ve buna bağlı olarak **w** referans vektörü orjine taşınır. Daha sonra **w** referans vektörü (3.1.2)



Şekil 3.2. $\mathbf{m}_A$, A sınıfındaki örneklerin merkezi olmak üzere, Kohonen vektörünün merkezi $\mathbf{c}$, yarıçapı $|\mathbf{cw}|$ olan hiper-küre üzerinde dönmesi ile oluşturulan öğrenme algoritması



Şekil 3.3. $\mathbf{w}(t+1)$ noktasının merkezi $\mathbf{p}$ ve yarıçapı $|\mathbf{pw}|$ olan hiper-küre içinde yer aldığı durum. $|\mathbf{m}_A\mathbf{w}(t+1)| < |\mathbf{m}_A\mathbf{w}(t)|$ ve $|\mathbf{pw}(t+1)| < |\mathbf{pw}|$ olduğundan dönme sonucunda öğrenme garanti altına alınır.

ve (3.1.3) eşitliklerinde görüldüğü üzere küresel koordinatlar cinsinden yazılır.

$$\left.\begin{array}{rcl} w_1 &=& \rho\cos(\alpha_1), \\ w_2 &=& \rho\sin(\alpha_1)\cos(\alpha_2), \\ &\vdots& \\ w_{n-1} &=& \rho\sin(\alpha_1)\sin(\alpha_2)\cdots\sin(\alpha_{n-2})\cos(\alpha_{n-1}) \end{array}\right\} \tag{3.1.2}$$

ve

$$w_n = \rho\sin(\alpha_1)\sin(\alpha_2)\cdots\sin(\alpha_{n-2})\sin(\alpha_{n-1}). \tag{3.1.3}$$

Eğer $\mathbf{w}$ vektörü α_1 ekseni etrafında $\Delta\alpha$ radyanı kadar döndürülürse, $\mathbf{w}(t+1)$ gibi yeni bir noktaya taşınır ve bu noktada t açısal koordinatlardaki iterasyon sayısıdır. s dönme yönü olmak üzere $\alpha_1 = \alpha_1 - s\Delta\alpha$ eşitliği geçerlidir. Eğer $s = -1$ ise, $\mathbf{w}$ vektörü α_1 ekseni etrafında saat yönünde döndürülür, $s = +1$ olduğu durumda ise saat yönünün aksine döndürme işlemi gerçekleşir. Döndürme işlemi tamamlandıktan sonra referans hiper-küre eski merkezine taşınır ve buna bağlı olarak $\mathbf{w}$ referans vektörü de yeni koordinatına taşınır.

$\mathbf{w}$ referans vektörünü referans hiper-küre üzerinde döndürülebilmesi için, sadece α_1 açısal koordinatını değiştirmek yeterli olacaktır, diğer açısal koordinatlar α_1 açısal koordinatının değişimine bağlı olarak yeniden elde edilir. Ancak bu durumda kaç derece döndürülmesi gerektiği akla gelmektedir. Açının hesaplanması için $\mathbf{p}, \mathbf{m}_A$ ve $\mathbf{w}(t)$ vektörleri farklı doğrultuda olduğunda, λ öğrenme katsayısı, $\beta_1 = \arccos\left(\dfrac{\|\mathbf{w}(t)\|}{w_1}\right)$ ve $\beta_2 = \arccos\left(\dfrac{p_1 - c_1}{\|\mathbf{cw}(t)\| - \|\mathbf{pw}(t)\|}\right)$ olmak üzere, dönme açısı (3.1.4) eşitliği kullanılarak hesaplanır.

$$\alpha_1 = \alpha_1 + \lambda|\beta_2 - \beta_1| \tag{3.1.4}$$

$\mathbf{w}(t+1)$ dairelerin bir kesişim noktasıdır ve $\mathbf{w}(t+1)$ içeren dairelerin parametrik eşitliği küresel koordinatlarda geçerlidir. Böylece kazanan vektör için yer değiştirme işlemi hiper-küre üzerinde gerçekleşmektedir ve $\mathbf{w}(t+1)$ referans vektörünün girdi vektörü $\mathbf{p}$ ve $\mathbf{m}_A$ ile aynı doğrultuda olmasından kaçınılmaktadır.

Eğer ters dönüşüm uygulanırsa $1 \leq i < n-1$ için, $\mathbf{w}$ vektörünün açısal koordinatları (3.1.5) ve (3.1.6) eşitlikleri ile tekrar hesaplanır.

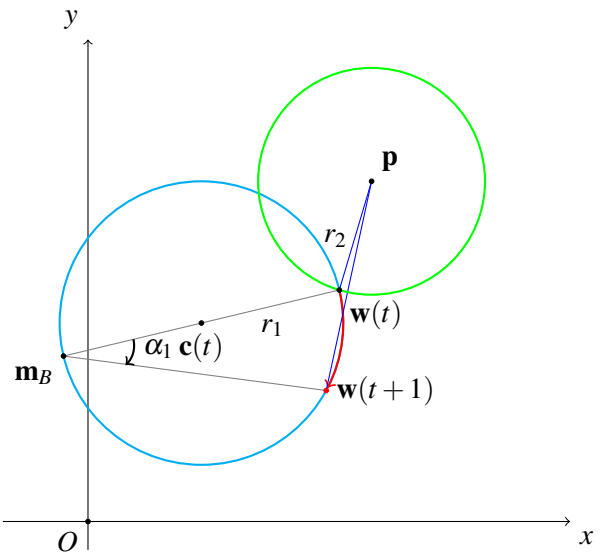
$$\alpha_i = \operatorname{arccot}\left(\frac{w_i}{\sqrt{\sum_{j=i+1}^{n} w_j^2}}\right) \tag{3.1.5}$$

$$\alpha_{n-1} = 2\operatorname{arccot}\left(\frac{\sqrt{w_{n-1}^2 + w_n^2} + w_{n-1}}{w_n}\right). \tag{3.1.6}$$

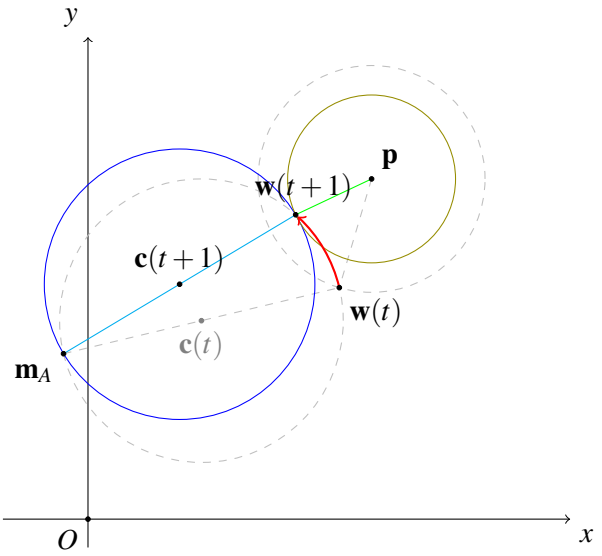
$\mathbf{w}(t+1)$ vektörünün öklidyen koordinatları, 3.1.2 ve 3.1.3 kullanılarak tekrar hesaplanır. Bu süreç ile, $\mathbf{w}$ vektörünün hem sınıf merkezine hemde girdi vektörüne yaklaşması garanti altına alınır. Şekil 3.3'te görüldüğü üzere, $\mathbf{w}(t+1)$ hiper-kürenin iç noktası olduğunda $|\mathbf{pw}(t+1)| < |\mathbf{pw}(t)|$ geçerlidir. Diğer yandan, uzun kiriş hiper-kürenin merkezinden geçtiği için $|\mathbf{m}_A\mathbf{w}(t+1)| < |\mathbf{m}_A\mathbf{w}(t)|$ sağlanır. Aynı zamanda, süreç boyunca çap değişmediği için, $\mathbf{w}(t+1)$ vektörü yine, merkezi $\mathbf{c}$ olan hiper-küre üzerinde yer almaktadır.

Önerilen metoda göre, eğer $\mathbf{w}$ global kazanan ise, girdi vektörü $\mathbf{p}$'den ve A sınıfının merkezinden uzaklaşmalı ve B sınıfının merkezine yaklaşmalıdır. Bundan dolayı Şekil 3.4'te görüldüğü üzere, global kazanan $\mathbf{w}$, merkezi $\mathbf{w}$ ile B sınıfının merkezinin orta noktası ve yarıçapı $|\mathbf{cw}|$ olan hiper-küre üzerinde döndürme işlemi gerçekleşmektedir. Bu durumda dönme sonucunda, $\mathbf{w}(t+1)$, merkezi girdi vektörü $\mathbf{x}$ ve yarıçapı $|\mathbf{pw}|$ olan hiper-kürenin dışında kalacaktır. Öğrenme yine garanti altına alınmaktadır, çünkü $|\mathbf{m}_B\mathbf{w}(t+1)| < |\mathbf{m}_B\mathbf{w}(t)|$ ve $|\mathbf{pw}(t+1)| > |\mathbf{pw}(t)|$ olduğu bilinmektedir.

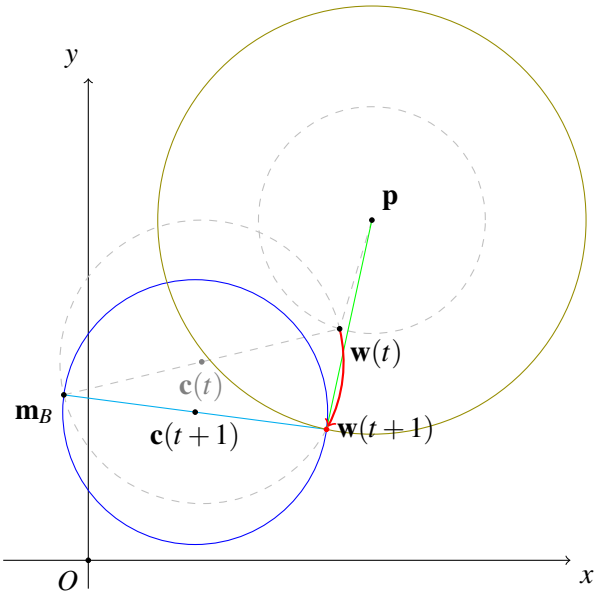
Bu süreç hem Şekil 3.5 ve hemde Şekil 3.6'da görüldüğü üzere öğrenme tamamlanıncaya veya maksimum iterasyon sayısına ulaşıncaya kadar aralıksız devam eder. Her bir iterasyonda, $\mathbf{w}$ vektörünün yerel ya da global kazanan vektör olup olmadığına bağlı olarak yakınsaklık yarıçapını belirlemek için iki tane hiper-küre oluşturulur. Bu yolla öğrenme kuralı her bir iterasyonda, kazanan sınıfın sınırlarının daralması veya genişlemesi işlemi gerçekleştirir.



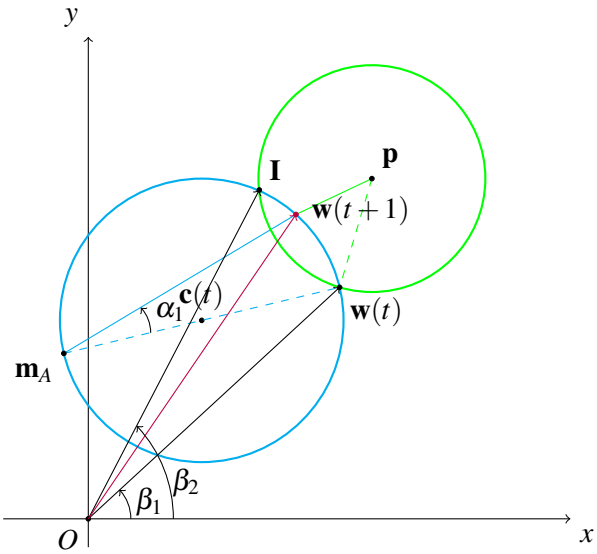
Şekil 3.4. $\mathbf{w}(t+1)$ noktasının merkezi $\mathbf{p}$ ve yarıçapı $|\mathbf{pw}|$ olan hiper-kürenin dışında yer aldığı durum. $|\mathbf{m}_B\mathbf{w}(t+1)| < |\mathbf{m}_B\mathbf{w}(t)|$ ve $|\mathbf{pw}(t+1)| > |\mathbf{pw}(t)|$ olduğundan, dönme sonucunda öğrenme garanti altına alınır



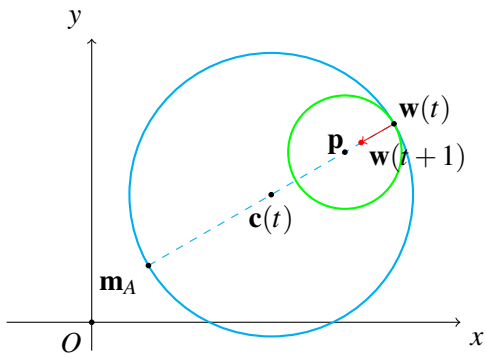
Şekil 3.5. $\mathbf{w}(t)$ yerel kazanan ise, girdi vektörü $\mathbf{p}$'ye ve A sınıfının merkezi $\mathbf{m}_A$'ya yaklaşır. $|\mathbf{m}_A\mathbf{w}(t+1)| < |\mathbf{m}_A\mathbf{w}(t)|$ ve $|\mathbf{pw}(t+1)| < |\mathbf{pw}(t)|$ olduğundan, dönme sonucunda öğrenme garanti altına alınır



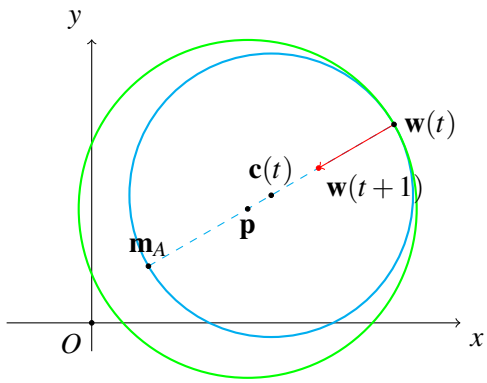
Şekil 3.6. $\mathbf{w}(t)$ global kazanan ise, girdi vektörü $\mathbf{p}$'den uzaklaşırken, B sınıfının merkezi $\mathbf{m}_B$'ye yaklaşmalıdır. $|\mathbf{m}_B\mathbf{w}(t+1)| < |\mathbf{m}_B\mathbf{w}(t)|$ ve $|\mathbf{p}\mathbf{w}(t)| < |\mathbf{p}\mathbf{w}(t+1)|$ olduğundan, dönme sonucunda öğrenme garanti altına alınır



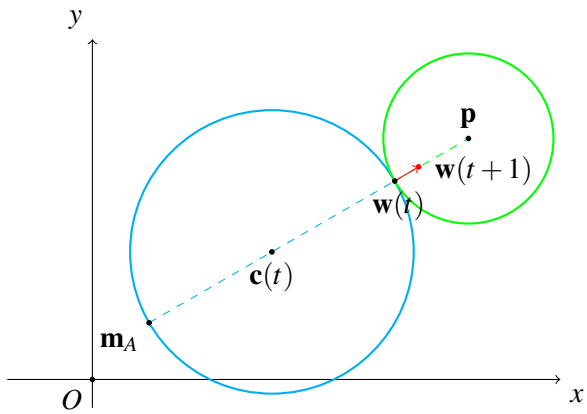
Şekil 3.7. Eğer üç nokta farklı doğrultuda olursa, dönme yönü ve dönme açısı λ öğrenme katsayısı olmak üzere $\Delta\alpha = \lambda|\beta_2 - \beta_1|$ olmalıdır



Şekil 3.8. Girdi vektörü **p** ve referans vektörü **w** aynı doğrultuda, $|\mathbf{pw}| < |\mathbf{cw}|$ ve hiper-küreler **w** koordinatlarında iç teğet ise, genelleştirilmiş delta öğrenme kuralı uygulanır



Şekil 3.9. Girdi vektörü **p** ve referans vektörü **w** aynı doğrultuda, $|\mathbf{pw}| > |\mathbf{cw}|$ ve hiper-küreler **w** koordinatlarında iç teğet ise, genelleştirilmiş delta öğrenme kuralı uygulanır



Şekil 3.10. Girdi vektörü **p** ve referans vektörü **w** aynı doğrultuda, $|\mathbf{pw}| < |\mathbf{cw}|$ ve hiper-küreler **w** koordinatlarında dış teğet ise, genelleştirilmiş delta öğrenme kuralı uygulanır

$\mathbf{p}, \mathbf{m}_A$ ve $\mathbf{w}(t)$ noktaları düşük bir olasılıkla aynı doğrultuda olabilir. Bu durum sadece aynı örnek, vektör nicemleme ağına arka arkaya gösterildiğinde gerçekleşir ve bu koşul altında, Şekil 3.8, Şekil 3.9 ve Şekil 3.10'da verilen üç durum nadiren oluşabilir. Belirtilen durum oluştuğunda Kohonen tarafından önerilen geleneksel genelleştirilmiş delta öğrenme kuralı kullanılmaktadır. Gerçekleşen tüm durumlarda, hiper-küreler birbirlerine **w** noktasında teğettirler. Şekil 3.8 ve Şekil 3.9'da görüldüğü üzere, verilen çemberler iç teğettirler. Eğer $\mathbf{w}(t)$ yerel kazanan ise, genelleştirilmiş delta öğrenme kuralı uygulanır ve yeni $\mathbf{w}(t+1)$ noktası hem girdi vektörü **p**'ye hemde A sınıfının merkezine yaklaşır.

Diğer yandan, Şekil 3.10'da görüldüğü üzere, eğer hiper-küreler birbirlerine dıştan teğet ise, $\mathbf{w}(t)$ girdi vektörü **p**'ye yaklaşırken, $\mathbf{m}_A$ sınıfının merkezinden uzaklaşmaktadır. Bu ise, **w** ile **p** ve $\mathbf{m}_A$'nın aynı doğrultuda olmalarından kaçınılması sebebidir.

Önerilen algoritmanın çalışma prensibini açıklamak üzere bir nümerik örnek inceleyelim. Nümerik örnekte önerilen öğrenme algoritmasının verimliliğini incelemek amacıyla, O_i, $i.$ girdi vektörü **p** için ağın ürettiği çıktı değeri ve T_i, $i.$ girdi vektörü için beklenen çıktı değeri olmak üzere, Eşitlik 3.1.7'de verilen

ortalama karesel hata miktarı (Mean Squared Error-MSE) kriteri kullanılmıştır. Bu kriteri kullanırken amaç hatanın en aza indirgenmesidir.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(O_i - T_i)^2 \tag{3.1.7}$$

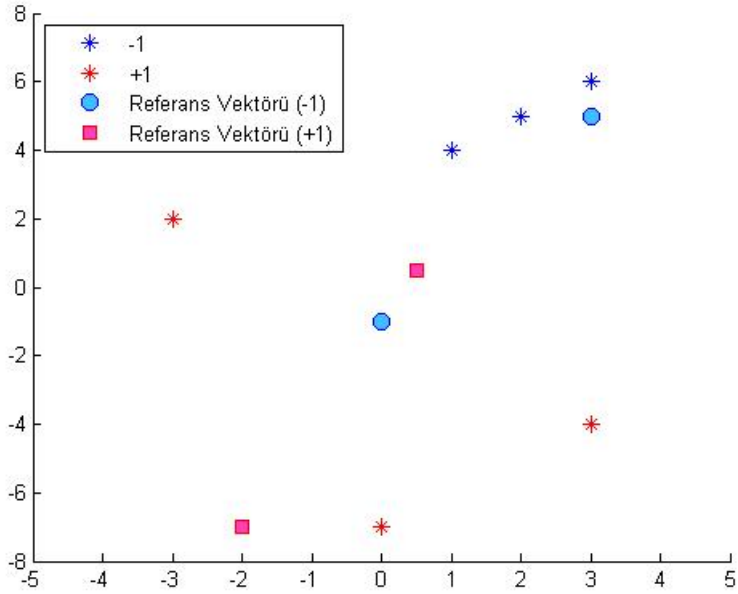
Örnek 3.1 Şekil 3.11 ile verilen iki farklı sınıf için $\mathbb{R}^2$ üzerinde alınan örneklem kümesi $P = \left\{ \begin{pmatrix} 1 \\ 4 \end{pmatrix}, \begin{pmatrix} 2 \\ 5 \end{pmatrix}, \begin{pmatrix} 3 \\ 6 \end{pmatrix}, \begin{pmatrix} 0 \\ -7 \end{pmatrix}, \begin{pmatrix} -3 \\ 2 \end{pmatrix}, \begin{pmatrix} 3 \\ -4 \end{pmatrix} \right\}$ olsun. P kümesindeki her bir örneğin ait olduğu sınıflar ise $T = \{1,1,1,\ -1,-1,-1\}$ kümesi ile belirlensin. Vektör nicemleme metodunun başlangıcında referans vektörlerinin ilk ağırlıkları $\mathbf{w}_1 = \begin{pmatrix} 3 \\ 5 \end{pmatrix}$, $\mathbf{w}_2 = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$, $\mathbf{w}_3 = \begin{pmatrix} -2 \\ -7 \end{pmatrix}$ ve $\mathbf{w}_4 = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$ olsun.

Verilen iki farklı sınıf için sınıf merkezleri ise $\mathbf{m}_{+1} = \begin{pmatrix} 2 \\ 5 \end{pmatrix}$ ve $\mathbf{m}_{-1} = \begin{pmatrix} 0 \\ -3 \end{pmatrix}$ olarak elde edilir.

Eğitim aşamasında kazanan nöronu belirlemek için girdi vektörü ile referans vektörleri arasındaki mesafeye bakılır. Girdi vektörüne en yakın olan nöron bu vektörle aynı sınıfta ise yerel kazanan; aksi taktirde global kazanan nöron olarak belirlenir. Tespit edilen yerel ve global kazanan nöronlar girdi vektörüne yakınlaştırılıp uzaklaştırılarak öğrenme gerçekleştirilir. Bu işlem istenilen oranda öğrenme sağlanana kadar her iterasyonda tekrar edilir.

Birinci İterasyon: İlk iterasyon aşamasında örneklem kümesindeki $\mathbf{p}_1 = \begin{pmatrix} 1 \\ 4 \end{pmatrix}$ vektörünü girdi vektörü olarak seçelim, $\mathbf{p} = \mathbf{p}_1$. Daha sonra referans vektörleri arasındaki uzaklıkları tespit edelim. İlk girdi vektörü $+1$ ile etiketlenmiş sınıfa ait olduğundan yerel kazanan nöron, $\mathbf{w}_3$ ve $\mathbf{w}_4$ referans vektörleri içerisinden $\mathbf{p}$ vektörüne en yakın olan nöron olacaktır.

Global kazanan nöron ise tüm referans vektörleri içerisinden $\mathbf{p}$ girdi vektörüne en yakın olan nörondur. Buna göre, $\mathbf{p}$ ile $\mathbf{w}_1$ arasındaki mesafe $|\mathbf{p}\mathbf{w}_1| = 2.2361$ olarak bulunur. Benzer şekilde $|\mathbf{p}\mathbf{w}_2| = 5.0990$, $|\mathbf{p}\mathbf{w}_3| = 11.4018$ ve $|\mathbf{p}\mathbf{w}_4| = 3.5355$ elde



Şekil 3.11. Örnek *3.1* için örneklem uzayı ve referans vektörleri

edilir. Buna göre, yerel kazanan nöron $\mathbf{w}_4$ ve global kazanan nöron ise $\mathbf{w}_1$ olarak tespit edilir.

Yerel kazanan nöron olan $\mathbf{w}_4$ vektörü $\sum_{i=1}^{n}(x_i - c_i)^2 = |\mathbf{cw}|$ ve $\sum_{i=1}^{n}(x_i - p_i)^2 \leq |\mathbf{pw}|$ eşitsizliklerini sağlaması gerektiğinden dönme yönü saat yönünde, $s = -1$ olarak alınmalıdır. Yerel kazanan vektörün ağırlığını güncelleme aşamasında öncelikle oluşan hiper-kürelerden yarıçapı $|\mathbf{cw}|$ olan hiper-kürenin merkezini orijine taşıyalım. O halde $\mathbf{c} = \begin{pmatrix} 1.25 \\ 2.75 \end{pmatrix}$ noktasından orjine taşınması durumunda $\mathbf{w}_4$ vektörü ise, $\mathbf{w}_4 = \begin{pmatrix} -0.75 \\ -2.25 \end{pmatrix}$ noktasına taşınır. Bu durumda hiper-kürelerin x ekseniyle yaptıkları açılar β_1 ve β_2 değerleri sırasıyla *(3.1.8)* ve *(3.1.9)* eşitliklerinde verilen formüller aracılığı ile hesaplanır.

$$\beta_1 = \arccos \frac{w_{41}}{|\mathbf{w}|} \tag{3.1.8}$$

$$\beta_{21} = \arccos \frac{c_1}{|\mathbf{c}|}\,, \quad \beta_{22} = \arccos \frac{p_1}{|\mathbf{p}|}\,, \quad \beta_2 = \max\{\beta_{21}, \beta_{22}\} \tag{3.1.9}$$

Buna göre, $\beta_1 = 1.8925$ radyan ve β_2 değeri ise, $\beta_{21} = 1.5708$ radyan ve $\beta_{22} = 1.3258$ radyan değerlerinin maksimumu seçilerek $\beta_2 = 1.5708$ radyan olarak hesaplanır. Ayrıca iki hiper-küre arasındaki açı α_1, $\mathbf{w}_{41} = \rho_1 \cos(\alpha_1)$ olduğundan Eşitlik 3.1.10 ile elde edilir.

$$\alpha_1 = \arccos\left(\frac{w_{41}}{\rho}\right) \tag{3.1.10}$$

O halde $\alpha_1 = 1.8925$ radyan olarak belirlenir ve buna bağlı olarak α_1 değeri için Eşitlik 3.1.11 kullanılmaktadır.

$$\alpha_1 = \alpha_1 + s\lambda|\beta_1 - \beta_2| \tag{3.1.11}$$

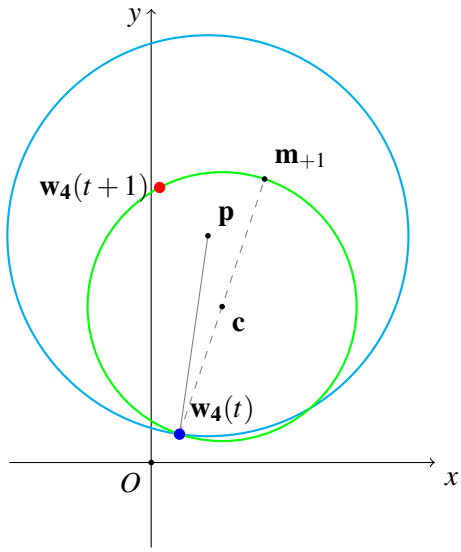
Böylece $\alpha_1 = 2.0354$ radyan olarak elde edilir. Ağırlık güncellemesi işlemi sonrasında elde edilen $\mathbf{w_4}$ değeri için Eşitlik 3.1.12 ve Eşitlik 3.1.13 değerleri kullanılmaktadır.

$$w_{41} = \rho \cos(\alpha_1) \tag{3.1.12}$$

$$w_{42} = \rho \sin(\alpha_1) \tag{3.1.13}$$

Yerel kazanan nöron olan $\mathbf{w}_4$ vektörü için ağırlık güncelleme işlemi sonrasında öteleme işleminin tersi yapılır ve $\mathbf{w}_4 = \begin{pmatrix} 0.1493 \\ 4.8508 \end{pmatrix}$ olarak bulunur. Şekil 3.12 ile görüldüğü üzere, lokal kazanan nöron $\mathbf{w_4}$ hem girdi vektörüne hemde temsil ettiği sınıf merkezine yaklaşmıştır.

Global kazanan nöron $\mathbf{w}_1$ için ise girdi vektöründen uzaklaştırma yapılması gerekir. Buna göre yerel kazanan nöron ile yapılan işlemler benzer şekilde $\mathbf{w}_1$ içinde yapılır. Öncelikle yarıçapı $|\mathbf{cw}|$ olan hiper-kürenin merkezini orjine taşıyalım. Dolayısıyla global kazanan nöron $\mathbf{w}_4 = \begin{pmatrix} 0.5 \\ 0 \end{pmatrix}$ noktasına taşınır. İlk iterasyona benzer şekilde işlemler yeni elde edilen referans vektörü için yapılır ve öteleme işleminin tersi uygulanarak ağırlık güncelleme işlemi sonlandırılır. Bu aşamada dönme yönü ise saat yönünün tersi $s = +1$ olarak belirlenir. Oluşan hiper-kürelerin x ekseniyle yaptığı açılar sırasıyla $\beta_1 = 0$ radyan ve β_2 değeri ise $\beta_{21} = 1.5708$ radyan

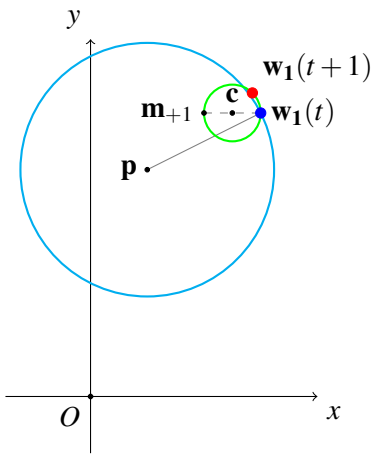


Şekil 3.12. Örnek *3.1* için eğitim aşamasında birinci iterasyonda oluşturulan referans hiper-küreler ve yerel kazanan nöronun güncellenmesi

ve $\beta_{22} = 1.3258$ radyan değerlerinin maksimumu seçilerek $\beta_2 = 1.5708$ radyan olarak hesaplanır. Ayrıca iki hiper-küre arasındaki açı α_1, Eşitlik 3.1.10 eşitliği yardımıyla $\alpha_1 = 0$ olarak hesaplanır. Buna bağlı olarak öğrenme katsayısı $\lambda = 0.5$ olmak üzere *(*3.1.11*)* eşitliği aracılığı ile $\alpha_1 = 0.7854$ radyan olarak hesaplanır. Elde edilen değerlere göre $\mathbf{w}_1$ vektörünün güncellenmesi sonrasında ve ardından öteleme işleminin tersi yapılırsa $\mathbf{w}_4 = \begin{pmatrix} 2.8536 \\ 5.3536 \end{pmatrix}$ olarak bulunur. Şekil 3.13 ile görüldüğü üzere, global kazanan nöron $\mathbf{w_1}$ girdi vektöründen uzaklaşmıştır. Yerel kazanan nöronun yakınlaştırılıp, global kazanan nöronun uzaklaştırılması sonucu eğitilen ağın performansını gözlemlemek amacıyla Eşitlik *3.1.7* ile verilen ortalama karesel hata miktarı kriteri $\mathbf{MSE}_1 = 3$ olarak hesaplanır. Öğrenme katsayısı monoton azalan bir değer olarak seçildiği için, ilk iterasyon sonucunda $\varepsilon = 0.001$ olmak üzere, *(*3.1.14*)* eşitliği kullanılarak $\lambda = 0.4995$ bulunur.

$$\lambda = \lambda - \varepsilon\lambda \tag{3.1.14}$$

İkinci İterasyon: İkinci iterasyonda $\mathbf{p}_2 = \begin{pmatrix} 2 \\ 5 \end{pmatrix}$ vektörü göz önünde bulundurulursa, girdi vektörü sınıf merkeziyle aynı olduğundan döndürme işlemi



Şekil 3.13. Örnek *3.1* için eğitim aşamasında birinci iterasyonda oluşturulan referans hiper-küreler ve global kazanan nöronun güncellenmesi

yapılmamaktadır. Dolayısıyla ortalama karesel hata miktarı için birinci iterasyonda hesaplanan değer geçerli olmaktadır. O halde $\mathbf{MSE}_1 = \mathbf{MSE}_2 = 3$ olarak hesaplanır.

Üçüncü İterasyon: Üçüncü iterasyonda ise $\mathbf{p}_3 = \begin{pmatrix} 3 \\ 6 \end{pmatrix}$ girdi vektörü için ağı eğitelim, $\mathbf{p} = \mathbf{p}_3$. Girdi vektörü ile referans vektörlerin arasındaki mesafeleri tespit edecek olursak, $\mathbf{p}$ ile $\mathbf{w}_1$ arasındaki mesafe $|\mathbf{pw}_1| = 0.6628$ ve diğer uzaklıklar sırasıyla $|\mathbf{pw}_2| = 7.6158$, $|\mathbf{pw}_3| = 13.9284$ ve $|\mathbf{pw}_4| = 11.2388$ olarak elde edilir. Buna göre, yerel kazanan nöron $\mathbf{w}_4$ ve global kazanan nöron ise $\mathbf{w}_1$ olarak bulunur. Öncelikle yerel kazanan referans vektörü için birinci iterasyonda bahsedilen dönme yönünün tespiti üçüncü iterasyonda benzer şekilde yorumlanır ve $s = -1$ olarak alınmaktadır. Hiper-kürenin merkezinin orjine taşınması sonrasında $\mathbf{w}_4 = \begin{pmatrix} -0.9254 \\ -0.0746 \end{pmatrix}$ elde edilir. Oluşan hiper-kürelerin x ekseniyle yaptıkları açılar β_1 ve β_2 değerleri *(3.1.8)* ve *(3.1.9)* eşitlikleri kullanılarak sırasıyla $\beta_1 = 3.0612$ radyan ve β_2 değeri ise $\beta_{21} = 1.5708$ radyan ve $\beta_{22} = 1.1071$ radyan değerlerinin maksimumu belirlenerek $\beta_2 = 1.5708$ radyan olarak tespit edilir. Ayrıca iki hiper-küre arasındaki α_1 açısı güncellenerek $\alpha_1 = 3.0612$ radyan olarak bulunur. Bu değer kullanılarak Eşitlik 3.1.11 aracılığıyla $\alpha_1 = 3.8057$ olarak bulunur.

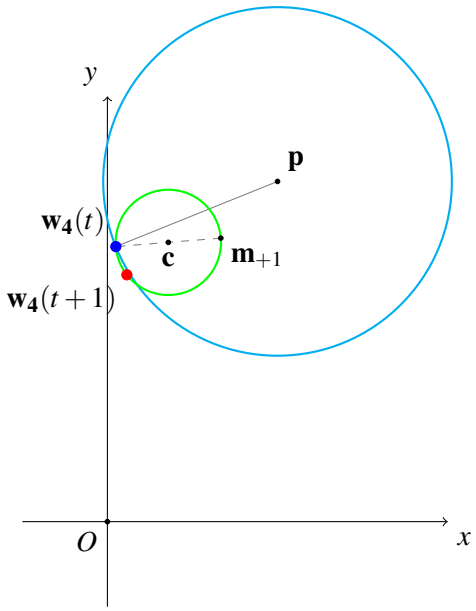
Referans vektörü için ağırlık güncellemesi ve öteleme işlemi tersi uygulanırsa $\mathbf{w}_4$ referans vektörünün bileşenleri $\mathbf{w}_4 = \begin{pmatrix} 0.3436 \\ 4.3532 \end{pmatrix}$ bulunur. Şekil 3.14 ile görüldüğü üzere, lokal kazanan nöron $\mathbf{w_4}$ hem girdi vektörüne hemde temsil ettiği sınıf merkezine yaklaşmıştır.

Global kazanan nöron $\mathbf{w}_1$ için benzer işlemler yapılırsa, öncelikle dönme yönü saat yönünde $s = -1$ olarak belirlenir. Yarıçapı **cw** olan hiper-kürenin merkezinin orjine taşınması sonucu global kazanan nöron da ötelenir ve $\mathbf{w}_1 = \begin{pmatrix} 0.4268 \\ 0.1768 \end{pmatrix}$ olarak tespit edilir. Oluşan hiper-kürelerin x ekseniyle yaptığı açılar sırasıyla $\beta_1 = 0.3929$ radyan ve $\beta_2 = 1.5708$ radyan, α_1 açısının ilk değeri $\alpha_1 = 0.3929$ radyan ve güncelleme sonucunda elde edilen değeri ise $\alpha_1 = 0.9224$ radyandır. Referans vektörünün ağırlığının günceleme aşaması ve öteleme işleminin tersinin uygulanması sonrasında $\mathbf{w}_1 = \begin{pmatrix} 2.7058 \\ 5.5450 \end{pmatrix}$ olarak güncellenir. Şekil 3.15 ile görüldüğü üzere, global kazanan nöron $\mathbf{w_1}$ girdi vektöründen uzaklaşmıştır. Eğitilen ağın performansını gözlemlemek amacıyla Eşitlik *3.1.7* ile verilen ortalama karesel hata miktarı kriteri $\mathbf{MSE}_3 = 2$ olarak hesaplanır. Öğrenme katsayısı monoton azalan bir değer olarak seçildiği için, üçüncü iterasyon sonucunda $\varepsilon = 0.001$ olmak üzere, *(*3.1.14*)* eşitliği kullanılarak $\lambda = 0.4990$ bulunur.

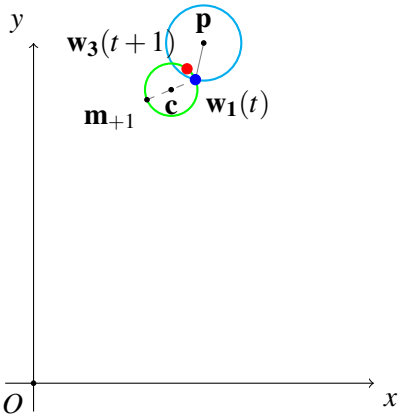
Dördüncü İterasyon: Dördüncü iterasyonda benzer şekilde $\mathbf{p}_4 = \begin{pmatrix} 0 \\ -7 \end{pmatrix}$ girdi vektörü için ağ eğitilir, $\mathbf{p} = \mathbf{p}_4$. Girdi vektörü ile referans vektörlerin arasındaki mesafeler

$|\mathbf{pw}_1| = 12.8335$, $|\mathbf{pw}_2| = 6$, $|\mathbf{pw}_3| = 2$ ve $|\mathbf{pw}_4| = 11.3584$ olarak elde edilir. Buna göre, yerel kazanan nöron $\mathbf{w}_2$ ve global kazanan nöron $\mathbf{w}_3$ referans vektörleridir. Bu iterasyonda yerel kazanan nöron için ağırlık güncelleme işlemi, girdi vektörü ve referans vektörü aynı doğrultuda olduğu için, *(*3.1.15*)* eşitliği ile verilen genelleştirilmiş delta öğrenme kuralı kullanılarak gerçekleştirilir.

$$\mathbf{w}_k(t+1) = \mathbf{w}_k(t) \mp \lambda(t)(\mathbf{x} - \mathbf{w}(t)) \tag{3.1.15}$$



Şekil 3.14. Örnek *3.1* için eğitim aşamasında üçüncü iterasyonda oluşturulan referans hiper-küreler ve yerel kazanan nöronun güncellenmesi



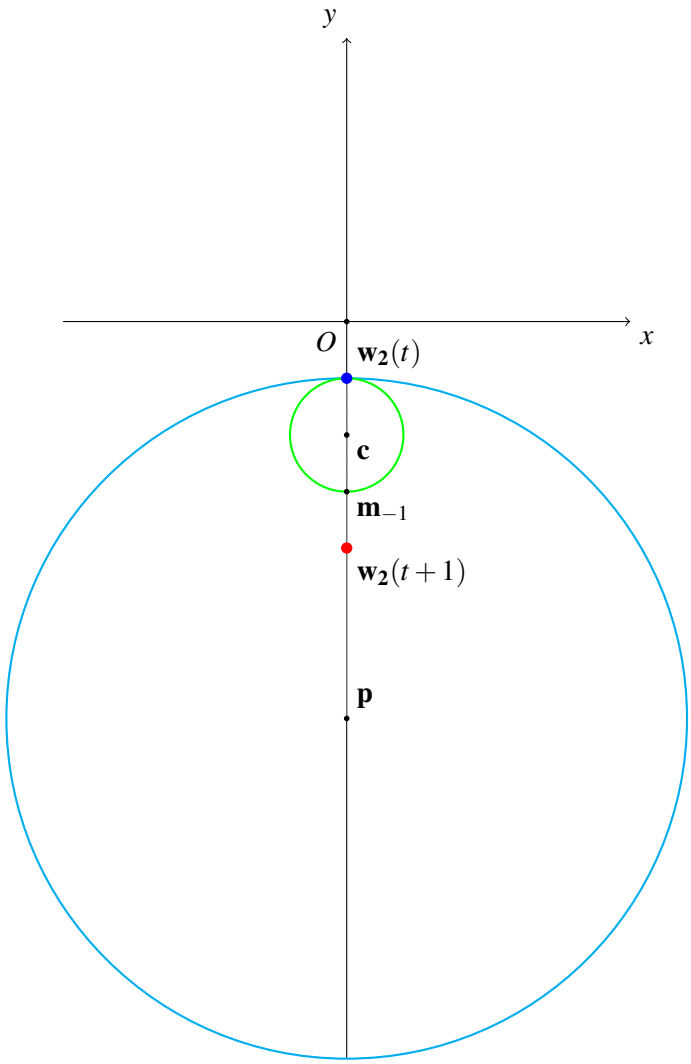
Şekil 3.15. Örnek *3.1* için eğitim aşamasında üçüncü iterasyonda oluşturulan referans hiper-küreler ve global kazanan nöronun güncellenmesi

Bu durumda ağırlık güncelleme işlemi sonrasında $\mathbf{w}_2 = \begin{pmatrix} 0 \\ -3.9940 \end{pmatrix}$ olarak bulunur. Şekil 3.16 ile görüldüğü üzere, lokal kazanan nöron $\mathbf{w_2}$ hem girdi vektörüne hemde temsil ettiği sınıf merkezine yaklaşmıştır.

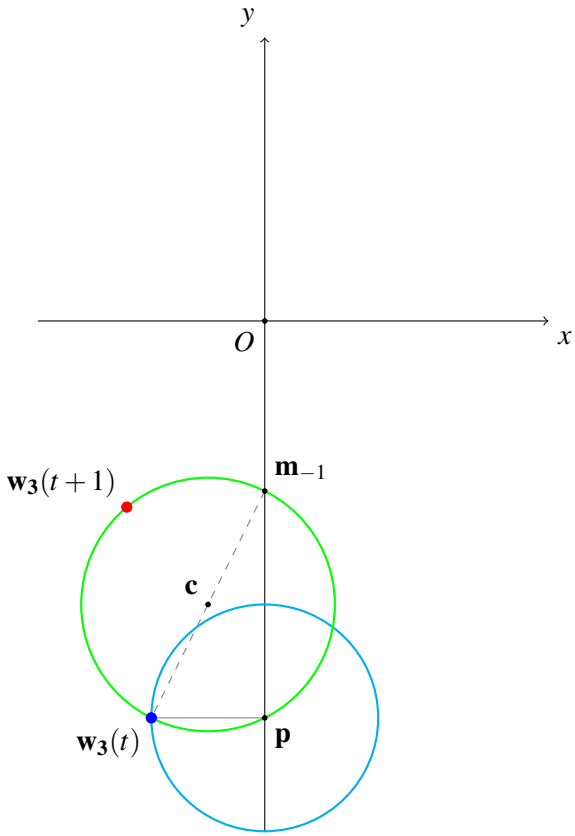
Global kazanan nöron için ise, dönme yönü saat yönünde $s = -1$ olarak tespit edilir. Hiper-kürenin orjine taşınması sonrasında referans vektörü $\mathbf{w}_3 = \begin{pmatrix} -1 \\ -2 \end{pmatrix}$ olarak bulunur. Buna göre oluşan hiper-küreler için elde edilen açılar ise sırasıyla x ekseniyle yaptığı açılar $\beta_1 = 2.0344$ radyan ve $\beta_2 = 1.5708$ radyan, α_1 açısının ilk değeri $\alpha_1 = 2.0344$ radyan ve bu değer güncellenerek $\alpha_1 = 2.2657$ radyan olarak hesaplanır. Referans vektörünün ağırlığının günceleme aşaması ve öteleme işleminin tersi uygulanması sonrasında $\mathbf{w}_3 = \begin{pmatrix} -2.4318 \\ -3.2824 \end{pmatrix}$ olarak güncellenir. Şekil 3.17 ile görüldüğü üzere, global kazanan nöron $\mathbf{w_3}$ girdi vektöründen uzaklaşmıştır. Eğitilen ağın performansını gözlemlemek amacıyla Eşitlik *3.1.7* ile verilen ortalama karesel hata miktarı kriteri $\mathbf{MSE}_4 = 3$ olarak hesaplanır. Öğrenme katsayısı dördüncü iterasyon sonucunda $\varepsilon = 0.001$ olmak üzere, *(*3.1.14*)* eşitliği kullanılarak $\lambda = 0.4985$ bulunur.

Verilen örnekte ortalama karesel hata miktarı 0 değerine ulaşınca ya da maksimum iterasyon sayısı kadar eğitim işlemi gerçekleşince ağın eğitilmesi sona erer.

t iterasyon sayısını belirtmek üzere, nümerik örnekte tüm iterasyonlarda yapılan ağırlık güncellemeleri α_{l_1}, α_{g_1} ve ortalama karesel hata miktarı (MSE) Çizelge 3.1 ile verilmiştir. Çizelge 3.1' de α_{l_1} lokal kazanan vektörün x ekseni etrafında radyan cinsinden ne kadar döndüğünü belirtir. Benzer şekilde α_{g_1} global kazanan vektörün x ekseni etrafında radyan cinsinden ne kadar döndüğünü belirtir. Çizelgede görüldüğü üzere, dördüncü iterasyonda genelleştirilmiş delta öğrenme kuralı uygulandığından dönme açısı hesaplanmamaktadır. α_{l_1} ve α_{g_1} belirlendikten sonra, lokal ve global kazanan vektörlerin küresel koordinat sisteminde diğer açısal koordinatlar *(3.1.5)* ve *(3.1.6)* eşitlikleri ile yeniden hesaplanır.



Şekil 3.16. Örnek *3.1* için eğitim aşamasında dördüncü iterasyonda oluşturulan referans hiper-küreler ve yerel kazanan nöronun güncellenmesi



Şekil 3.17. Örnek *3.1* için eğitim aşamasında dördüncü iterasyonda oluşturulan referans hiper-küreler ve global kazanan nöronun güncellenmesi

Çizelge 3.1. LVQRKV metodu kullanılarak çözüme ulaşılan nümerik örnek için ağırlık güncellemesi, α_{l_1}, α_{g_1} ve ortalama karesel hata miktarı (MSE)

LVQRKV							
t	$\mathbf{w}_1$	$\mathbf{w}_2$	$\mathbf{w}_3$	$\mathbf{w}_4$	α_{l_1}	α_{g_1}	**MSE**
0	(3.0000, 5.0000)	(0.0000, -1.0000)	(-2.0000, -7.0000)	(0.5000, 0.5000)			
1	(2.8536, 5.3536)	(0.0000, -1.0000)	(-2.0000, -7.0000)	(0.1493, 4.8508)	2.0354	0.7854	3.0000
2	(2.8536, 5.3536)	(0.0000, -1.0000)	(-2.0000, -7.0000)	(0.1493, 4.8508)	0.0000	0.0000	3.0000
3	(2.7058, 5.5450)	(0.0000, -1.0000)	(1.3919, 0.2504)	(0.3436, 4.3532)	3.8057	0.9224	2.0000
4	(2.7058, 5.5450)	(0.0000, -3.9940)	(-2.4318, -3.2824)	(0.3436, 4.3532)	——-	2.2657	2.0000

Tez çalışmasının bir sonraki bölümünde, önerilen öğrenme algoritmasının etkinliğini gözlemlemek ve performans analizi yapmak üzere, gerçekleştirilen deneysel çalışmalardan bahsedilecektir.

4. DENEYSEL ÇALIŞMALAR

4.1. Giriş

Tez çalışmasında, önerilen metodun geçerliliğini ölçmek amacıyla dört farklı veri seti kullanılarak deneysel çalışmalar yapılmıştır. Veri setlerinden üçü Kaliforniya Üniversitesi UCI Repository tarafından hazırlanan veri setinden temin edilmiştir [27]. Diğer veri seti ise bazı Türk köşe yazarlarının eserlerinden alıntı yapılarak, tarafımızdan oluşturulmuş doküman sınıflandırmadır. Bu veri setleri için herhangi bir boyut indirgeme yöntemi uygulanmamıştır. Tüm metotlar aynı topolojiye sahiptir. Yani, her bir yöntem için Kohonen katmanında yarışan nöronlar, aynı nöron sayısına sahip N_C sınıfına ayrılır ve bir çıktı katmanıyla eşleşirler.

Eğitim aşamasında, yarışmacı Kohonen katmanında ağırlıkların başlangıç değerleri, sınıf merkezlerinin koordinatları olarak alınmıştır. Bu yaklaşım sayesinde, tüm metotlar için aynı ağırlıklar kullanıldığından daha geçerli ve güvenilir karşılaştırma yapılmaktadır. Ayrıca, tüm ağlar için eğitim aşamasında, aynı örnekler aynı sırada ağa gösterilir. LVQRKV metodunun bilinen diğer LVQ yöntemleriyle karşılaştırma yapılabilmesi amacıyla, öğrenme oranı monoton azalan değerler olmak üzere başlangıçta 0.75, 0.5 and 0.25 olarak belirlenmiştir. Sabit dengeleyici faktör olan ε ise 10^{-3} olarak seçilmiştir. Tüm deneyler için maksimum iterasyon sayısı 300, bağıl pencere genişliği ise 0.2'dir ve sağlıklı bir karşılaştırma yapılabilmesi için 10-katlı çapraz doğrulama yöntemi kullanılmıştır.

10-katlı çapraz doğrulama yönteminde, öncelikle örneklem kümesi her sınıftan eşit sayıda örnek içerecek şekilde 10 altörneklem kümesine ayrılır. Tüm deneylerde, bir tane altörneklem kümesi ağın test edilmesi için kullanılır. Diğer örnekler ise, modeli eğitmek için kullanılır. Daha güvenilir sonuç elde etmek için, bu süreç 10 kez tekrarlanır. Deneysel çalışmamızın son aşamasında, 10 kez tekrar edilen deney sonuçlarına göre, (4.1.1) eşitliğinde verilen F-ölçüm değerleri ve (4.1.2)

eşitliğinde verilen doğruluk katsayısı skorlarının ortalaması ve standart sapması hesaplanmıştır ve elde edilen sonuçlar ilgili çizelgelerde sunulmuştur. F-ölçüm ve doğruluk skorlarını elde etmek için 4.1.3 ve 4.1.4 formülleriyle elde edilen hassasiyet ve duyarlılık skorları kullanılmaktadır.

$$\text{F-ölçüm (F-measure)} = \frac{2 \times \text{Hassasiyet} \times \text{Duyarlılık}}{\text{Hassasiyet} + \text{Duyarlılık}} \tag{4.1.1}$$

$$\text{Doğruluk (Accuracy)} = \frac{\text{TP+TN}}{\text{TP+TN+FP+FN}} \tag{4.1.2}$$

$$\text{Hassasiyet (Precision)} = \frac{\text{TP}}{\text{TP+FP}} \tag{4.1.3}$$

$$\text{Duyarlılık (Recall)} = \frac{\text{TP}}{\text{TP+FN}} \tag{4.1.4}$$

Performans ölçümlerinde kulllanılan değerler Çizelge 4.1 üzerinde görüldüğü üzere verilmiştir. TP değeri pozitif sınıftaki örneklerin kaç tanesinin sistem tarafından doğru sınıflandırıldığını gösterirken, FP değeri ise pozitif sınıftaki örneklerin kaçının sistem tarafından yanlış sınıflandırıldığını gösterir. Benzer şekilde TN ve FN değerleri ise, negatif sınıftaki örneklerin kaçının sistem tarafından sırasıyla doğru veya yanlış sınıflandırıldığını belirtir.

Çizelge 4.1. Hata Matrisi

	Sistem Doğru	Sistem Yanlış
Referans Doğru	TP	FN
Referans Yanlış	FP	TN

Çalışmada kullanılan veri setlerinin yapısı ve karmaşıklığının anlaşabilmesi için, Tekil Değer Ayrışımı (Singular Value Decomposition) yöntemi kullanılarak öznitelik vektör uzayının boyutu ikiye indirgenmiş ve örnekler Şekil 4.2, Şekil 4.1 ve Şekil 4.4 ile gösterilmiştir. Ancak bu işlem sadece örneklem uzayındaki örneklerin dağılımını görüntülemek amacıyla yapılmıştır. Vektör nicemleme yöntemlerinin ve tezde önerilen yaklaşımın karşılaştırılması aşamasında verisetlerinde bir boyut indirgeme işlemi uygulanmamıştır.

Tekil Değer Ayrışımı, örüntü tanıma ve makine öğreniminde çoğu zaman boyut indirgeme amacıyla kullanılan temel adımdır.

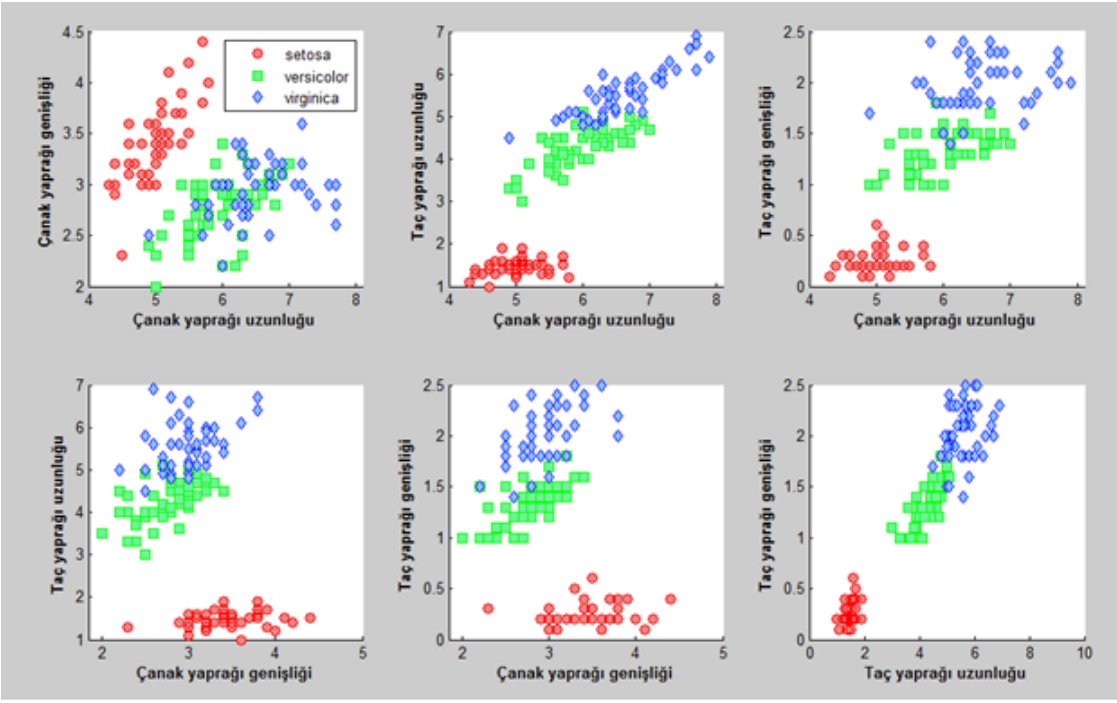
Tanım 4.1 *S,* $n \times n$ boyutunda bir köşegen matris*, U,* $m \times n$ *ve* V^t*,* $n \times n$ boyutlu ortogonal matrisler olmak üzere*, A* matrisinin $A = USV^t$ biçiminde üç ayrı matrisin çarpımına ayrıştırılabilme işlemine Tekil Değer Ayrışımı (Singular Value Decomposition) denir.

Tekil Değer Ayrışımı yöntemi aracılığıyla her bir sınıfa ait örneklerin dağılımını gözlemlemek mümkündür. Elde edilen Şekil 4.2, Şekil 4.1, Şekil 4.4 ve Şekil'den de anlaşılacağı üzere, sınıf sınırlarını belirlemek her zaman kolay değildir. Vektör Nicemleme Ağlarının tercih edilme sebebi sınıf sınırlarını belirlemeden sınıflandırma yapabilmesidir.

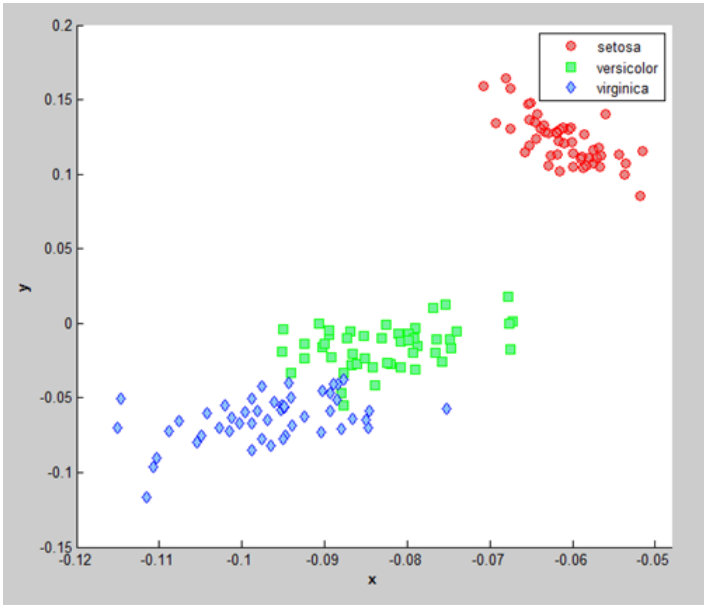
4.2. İris Veri Seti İçin Deneysel Sonuçlar

Tez çalışmasında gerçekleştirilen ilk deney, üç farklı iris çiçeği türü için oluşturulan ve toplamda 150 örnek içeren ve makine öğrenmesi alanında en sık kullanılan veri setidir [27, 29]. Her bir örnek, cm cinsinden çiçeğin çanak yaprağı uzunluğu ve genişliği, taç yaprağı uzunluğu ve genişliği olmak üzere dört bileşenden oluşan öznitelik vektörüne sahiptir. İris veri seti için, Tekil Değer Ayrışımı metodu ile öznitelik vektör uzayının boyutu ikiye indirilmiş ve örneklerin dağılımı Şekil 4.2 ile verilmiştir. Buna ek olarak, vektör bileşenlerinin her birinin birbirleri arasında olan dağılımı Şekil 4.1 üzerinde gösterilmiştir.

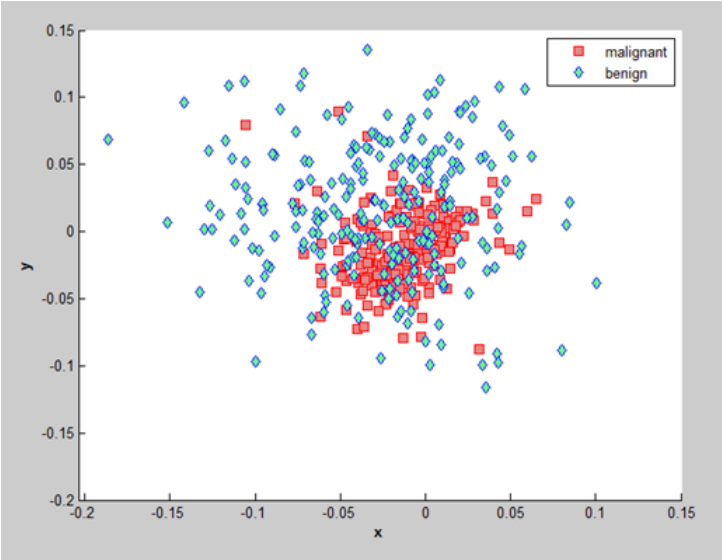
Oluşturulan vektör nicemleme ağının Kohonen katmanında, her sınıf 10 nörona sahiptir. Önerilen yöntem, farklı başlangıç değerlerine sahip monoton azalan öğrenme katsayıları kullanılarak, küçük veri setlerine uygulanmıştır. Sınıflandırmanın performansını gözlemlemek için Çizelge 4.9 oluşturulmuştur.



Şekil 4.1. İris veriseti için kullanılan öznitelik vektörünün bileşenlerinin birbirlerine göre dağılımı



Şekil 4.2. Tekil değer ayrışımı ile boyut indirgeme yöntemi uygulanan iris veriseti örneklem dağılımı



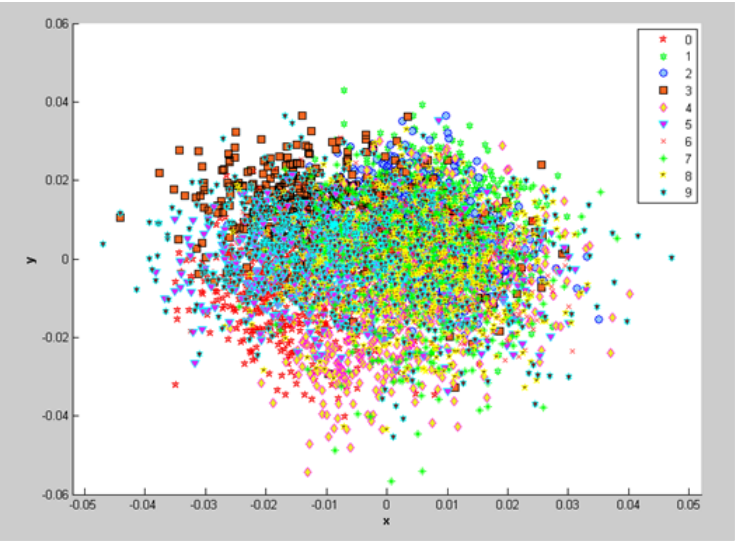
Şekil 4.3. Tekil değer ayrışımı ile boyut indirgeme yöntemi uygulanan Wisconsin meme kanseri veriseti örneklem dağılımı

4.3. Wisconsin Meme Kanseri Tespiti Veri Seti için Deneysel Sonuçlar

İkinci deneysel çalışmada, Wisconsin meme kanseri tespiti için Wisconsin Üniversitesi tarafından oluşturulan veri seti kullanılmıştır [27, 30]. Her bir örnek, meme ucundan aspirasyon işlemi ile çekilen sıvıların dijital görüntülerinden elde edilen verilerle hesaplanmış 32 bileşenli öznitelik vektörüne sahiptir. Kohonen katmanında ağ topolojisi 20 nörona sahiptir. Veriseti üzerinde önerilen öğrenme algoritmasının performansı ile LVQ ağlarının bazı çeşitleriyle Çizelge 4.4'de görüldüğü üzere karşılaştırılmıştır. Bununla birlikte, Wisconsin meme kanseri tespiti veri seti için örneklerin dağılımını göstermek amacıyla, Tekil Değer Ayrışımı metodu kullanılarak Şekil 4.3 kullanılmıştır.

4.4. Optik Rakam Tanıma Veri Seti için Deneysel Sonuçlar

Tez çalışmasında çözüm aranan sınıflandırma problemlerinden biri de daha karmaşık yapıya sahip olan optik rakam tanıma problemidir [27, 31, 32]. Veri seti 64 bileşenli öznitelik vektörüne sahip 5620 örnekten oluşmaktadır. Her bir sınıf



Şekil 4.4. Tekil değer ayrışımı ile boyut indirgeme yöntemi uygulanan optik rakam tanıma veriseti örneklem dağılımı

Kohonen katmanında 30 nöron içermektedir. Eğitim aşamasından sonra, sırasıyla 0.75, 0.5 ve 0.25 gibi farklı başlangıç değerlerine sahip öğrenme katsayıları ile elde edilen sonuçlar Çizelge 4.6'da verilmiştir. Aynı zamanda, optik rakam tanıma veri seti için örneklerin dağılımı Şekil 4.4 ile verilmiştir.

4.5. Doküman Sınıflandırma Veri Seti için Deneysel Sonuçlar

Son olarak Türkçe haberlerden alıntı yapılarak oluşturulan derlem, doküman sınıflandırma problemi için kullanılmıştır. Bu derlem 'ekonomi', 'sanat ve kültür', 'sağlık' ve 'spor' olmak üzere dört farklı kategoriden oluşmaktadır ve toplam 300 Türkçe doküman içermektedir. Önişleme aşamasında, noktalama işaretleri gibi tüm gereksiz karakterler dokümandan çıkarılmıştır. Büyük harflerin tümü küçük harfe dönüştürülmüş ve arka arkaya gelen iki sözcük arasında sadece tek bir boşluk olacak şekilde düzenlenmiştir. Sözcüklerin frekansı çıkarıldıktan sonra, özellik çıkarma aşamasında tüm dökumanlar için sözcük monogramları hesaplanır. Dökumanlar kelime torbası (bag-of-words) modeli kullanılarak sözcüklerin frekanslarını içeren sütun vektörleri olarak temsil

edilir. Tüm yöntemler için ağların Kohonen katmanında her bir sınıf için 50 nöron bulunmaktadır. Bu uygulamada her bir dökuman 400 öznitelik vektörü içerdiğinden, tez çalışmamızda yer verilen en karmaşık sınıflandırma problemidir. Öznitelik vektörü her bir sınıfta frekansı en yüksek 400 sözcük kullanılarak inşa edilir. Önerilen öğrenme algoritmasının performansını gözlemlemek amacıyla, oluşturulan derlem üzerinde önerilen öğrenme algoritması ve diğer LVQ ağları uygulanmıştır. Elde edilen sonuçlar Çizelge 4.8 ile verilmiştir. Doküman sınıflandırma veri seti için örneklerin dağılımı Tekil Değer Ayrışımı kullanılmasına rağmen karmaşık bir yapıda olduğundan bu veri seti için şekil ile sunulmaya gerek görülmemiştir.

Çizelge 4.2. Referans vektörlerinin ilk ağırlıklarının sınıf merkezleri olarak seçildiği ve monoton azalan öğrenme oranı ile 10-katlı çapraz doğrulama yöntemine göre iris veri seti üzerinde performans analizi

Öğrenme Oranının Başlangıç Değerleri	**Yöntem**	**Ortalama Doğruluk**	**Ortalama F-ölçüm**
0.25	LVQ1	**0.9778 ± 0.0612**	**0.9663 ± 0.0926**
	LVQ2	0.9556 ± 0.0691	0.9321 ± 0.1059
	LVQ2.1	0.9556 ± 0.0691	0.9321 ± 0.1059
	LVQ3	0.9630 ± 0.0533	0.9429 ± 0.0839
	OLVQ1	0.9333 ± 0.0804	0.8956 ± 0.1326
	OLVQ3	0.9333 ± 0.0804	0.8956 ± 0.1326
	GLVQ	0.9556 ± 0.0554	0.9314 ± 0.0874
	RLVQ	0.9704 ± 0.0768	0.9556 ± 0.1152
	GRLVQ	0.9630 ± 0.0533	0.9429 ± 0.0839
	LVQRKV	**0.9407 ± 0.0811**	**0.9071 ± 0.1336**
0.5	LVQ1	0.9556 ± 0.0691	0.9321 ± 0.1059
	LVQ2	**0.9778 ± 0.0452**	**0.9657 ± 0.0709**
	LVQ2.1	**0.9778 ± 0.0452**	**0.9657 ± 0.0709**
	LVQ3	0.9630 ± 0.0674	0.9435 ± 0.1030
	OLVQ1	0.9333 ± 0.0804	0.8956 ± 0.1326
	OLVQ3	0.9333 ± 0.0804	0.8956 ± 0.1326
	GLVQ	0.9407 ± 0.0699	0.9064 ± 0.1192
	RLVQ	0.9630 ± 0.0790	0.9441 ± 0.1191
	GRLVQ	0.9481 ± 0.0811	0.9185 ± 0.1335
	LVQRKV	**0.9556 ± 0.0691**	**0.9321 ± 0.1059**
0.75	LVQ1	**0.9630 ± 0.0674**	**0.9435 ± 0.1030**
	LVQ2	0.9481 ± 0.0699	0.9206 ± 0.1075
	LVQ2.1	0.9481 ± 0.0699	0.9206 ± 0.1075
	LVQ3	0.9481 ± 0.0811	0.9213 ± 0.1231
	OLVQ1	0.9333 ± 0.0804	0.8956 ± 0.1326
	OLVQ3	0.9333 ± 0.0804	0.8956 ± 0.1326
	GLVQ	0.9481 ± 0.0757	0.9182 ± 0.1265
	RLVQ	0.9556 ± 0.0691	0.9321 ± 0.1059
	GRLVQ	0.8963 ± 0.1128	0.8342 ± 0.1912
	LVQRKV	**0.9556 ± 0.0554**	**0.9314 ± 0.0874**

Çizelge 4.3. Referans vektörlerinin ilk ağırlıklarının rastgele seçildiği ve monoton azalan öğrenme oranı ile 10-katlı çapraz doğrulama yöntemine göre iris veri seti üzerinde performans analizi

Öğrenme Oranının Başlangıç Değerleri	**Yöntem**	**Ortalama Doğruluk**	**Ortalama F-ölçüm**
0.25	LVQ1	0.5778 ± 0.1736	0.2056 ± 0.2893
	LVQ2	0.6963 ± 0.1844	0.4218 ± 0.4015
	LVQ2.1	0.6963 ± 0.1890	0.4389 ± 0.4117
	LVQ3	0.7926 ± 0.1591	0.5886 ± 0.4145
	OLVQ1	0.7556 ± 0.1635	0.5127 ± 0.4181
	OLVQ3	0.8593 ± 0.1956	0.7740 ± 0.3303
	GLVQ	0.6741 ± 0.1725	0.4419 ± 0.3281
	RLVQ	0.9333 ± 0.1036	0.8875 ± 0.2109
	GRLVQ	$\mathbf{0.9630 \pm 0.0533}$	$\mathbf{0.9429 \pm 0.0839}$
	LVQRKV	$\mathbf{0.9333 \pm 0.0750}$	$\mathbf{0.8962 \pm 0.1223}$
0.5	LVQ1	0.5556 ± 0.1598	0.1667 ± 0.2397
	LVQ2	0.6667 ± 0.1957	0.3611 ± 0.3965
	LVQ2.1	0.7333 ± 0.1836	0.4778 ± 0.4238
	LVQ3	0.7333 ± 0.1789	0.4758 ± 0.4201
	OLVQ1	0.7111 ± 0.1765	0.4329 ± 0.4067
	OLVQ3	0.7852 ± 0.2163	0.6498 ± 0.3788
	GLVQ	0.7259 ± 0.2159	0.5152 ± 0.4158
	RLVQ	0.7704 ± 0.2333	0.5799 ± 0.4289
	GRLVQ	0.9481 ± 0.0699	0.9206 ± 0.1075
	LVQRKV	$\mathbf{0.9556 \pm 0.0554}$	$\mathbf{0.9314 \pm 0.0874}$
0.75	LVQ1	0.5704 ± 0.1591	0.2030 ± 0.2531
	LVQ2	0.6444 ± 0.1900	0.3202 ± 0.3739
	LVQ2.1	0.7259 ± 0.1907	0.4835 ± 0.4023
	LVQ3	0.6000 ± 0.1789	0.2425 ± 0.3205
	OLVQ1	0.6444 ± 0.2191	0.3385 ± 0.3870
	OLVQ3	0.7704 ± 0.2202	0.5765 ± 0.4141
	GLVQ	0.6370 ± 0.2673	0.4276 ± 0.3963
	RLVQ	0.6667 ± 0.2241	0.3802 ± 0.3983
	GRLVQ	0.9111 ± 0.0941	0.8464 ± 0.2119
	LVQRKV	$\mathbf{0.9259 \pm 0.0790}$	$\mathbf{0.8842 \pm 0.1306}$

Çizelge 4.4. Referans vektörlerinin ilk ağırlıklarının sınıf merkezleri olarak seçildiği ve monoton azalan öğrenme oranı ile 10-katlı çapraz doğrulama yöntemine göre Wisconsin meme kanseri tespiti veri seti üzerinde performans analizi

Öğrenme Oranının Başlangıç Değerleri	**Yöntem**	**Ortalama Doğruluk**	**Ortalama F-ölçüm**
0.25	LVQ1	0.9375 ± 0.0331	0.9373 ± 0.0336
	LVQ2	0.9542 ± 0.0314	0.9541 ± 0.0315
	LVQ2.1	0.9542 ± 0.0314	0.9541 ± 0.0315
	LVQ3	0.9583 ± 0.0253	0.9583 ± 0.0254
	OLVQ1	0.9437 ± 0.0318	0.9437 ± 0.0320
	OLVQ3	0.9521 ± 0.0254	0.9520 ± 0.0255
	GLVQ	0.9521 ± 0.0254	0.9520 ± 0.0254
	RLVQ	0.8979 ± 0.0510	0.8975 ± 0.0519
	GRLVQ	0.9625 ± 0.0249	0.9624 ± 0.0251
	LVQRKV	$\mathbf{0.9667 \pm 0.0274}$	$\mathbf{0.9666 \pm 0.0274}$
0.5	LVQ1	0.9479 ± 0.0257	0.9478 ± 0.0259
	LVQ2	0.9563 ± 0.0261	0.9562 ± 0.0262
	LVQ2.1	0.9563 ± 0.0261	0.9562 ± 0.0262
	LVQ3	0.9479 ± 0.0348	0.9478 ± 0.0350
	OLVQ1	0.9437 ± 0.0318	0.9437 ± 0.0320
	OLVQ3	0.9521 ± 0.0254	0.9520 ± 0.0255
	GLVQ	0.9479 ± 0.0335	0.9478 ± 0.0337
	RLVQ	0.9229 ± 0.0488	0.9226 ± 0.0498
	GRLVQ	$\mathbf{0.9646 \pm 0.0254}$	$\mathbf{0.9645 \pm 0.0257}$
	LVQRKV	$\mathbf{0.9646 \pm 0.0215}$	$\mathbf{0.9645 \pm 0.0216}$
0.75	LVQ1	0.9417 ± 0.0368	0.9415 ± 0.0372
	LVQ2	0.9604 ± 0.0278	0.9604 ± 0.0279
	LVQ2.1	0.9604 ± 0.0278	0.9604 ± 0.0279
	LVQ3	0.9583 ± 0.0253	0.9583 ± 0.0254
	OLVQ1	0.9417 ± 0.0355	0.9415 ± 0.0359
	OLVQ3	0.9521 ± 0.0254	0.9520 ± 0.0255
	GLVQ	0.9604 ± 0.0309	0.9604 ± 0.0310
	RLVQ	0.8854 ± 0.0526	0.8849 ± 0.0536
	GRLVQ	0.9104 ± 0.0741	0.9078 ± 0.0822
	LVQRKV	$\mathbf{0.9646 \pm 0.0254}$	$\mathbf{0.9645 \pm 0.0256}$

Çizelge 4.5. Referans vektörlerinin ilk ağırlıklarının rastgele seçildiği ve monoton azalan öğrenme oranı ile 10-katlı çapraz doğrulama yöntemine göre Wisconsin meme kanseri tespiti veri seti üzerinde performans analizi

Öğrenme Oranının Başlangıç Değerleri	**Yöntem**	**Ortalama Doğruluk**	**Ortalama F-ölçüm**
0.25	LVQ1	0.9500 ± 0.0525	0.9492 ± 0.0558
	LVQ2	0.6375 ± 0.1742	0.5387 ± 0.3579
	LVQ2.1	0.7208 ± 0.2046	0.6517 ± 0.3563
	LVQ3	0.9667 ± 0.0196	0.9666 ± 0.0198
	OLVQ1	0.7104 ± 0.1470	0.6600 ± 0.2561
	OLVQ3	0.8500 ± 0.0456	0.8473 ± 0.0533
	GLVQ	0.7563 ± 0.1293	0.7491 ± 0.1477
	RLVQ	0.6354 ± 0.1774	0.5531 ± 0.3416
	GRLVQ	0.9583 ± 0.0234	0.9582 ± 0.0238
	LVQRKV	$\mathbf{0.9646 \pm 0.0332}$	$\mathbf{0.9646 \pm 0.0332}$
0.5	LVQ1	0.9313 ± 0.0395	0.9305 ± 0.0420
	LVQ2	0.6646 ± 0.1850	0.5755 ± 0.3581
	LVQ2.1	0.6292 ± 0.1780	0.5233 ± 0.3699
	LVQ3	$\mathbf{0.9646 \pm 0.0235}$	$\mathbf{0.9645 \pm 0.0237}$
	OLVQ1	0.6250 ± 0.1734	0.5223 ± 0.3573
	OLVQ3	0.8438 ± 0.0913	0.8370 ± 0.1127
	GLVQ	0.8042 ± 0.1976	0.7931 ± 0.2346
	RLVQ	0.5250 ± 0.1154	0.4101 ± 0.3251
	GRLVQ	0.9354 ± 0.0309	0.9350 ± 0.0320
	LVQRKV	$\mathbf{0.9521 \pm 0.0332}$	$\mathbf{0.9520 \pm 0.0334}$
0.75	LVQ1	0.8250 ± 0.1876	0.7889 ± 0.2876
	LVQ2	0.6625 ± 0.2043	0.5660 ± 0.3788
	LVQ2.1	0.7104 ± 0.2022	0.6359 ± 0.3552
	LVQ3	$\mathbf{0.9625 \pm 0.0267}$	$\mathbf{0.9624 \pm 0.0271}$
	OLVQ1	0.5542 ± 0.2749	0.5090 ± 0.3491
	OLVQ3	0.8583 ± 0.1064	0.8480 ± 0.1486
	GLVQ	0.7979 ± 0.1509	0.7866 ± 0.1795
	RLVQ	0.6021 ± 0.1615	0.5200 ± 0.3180
	GRLVQ	0.8917 ± 0.1417	0.8791 ± 0.1842
	LVQRKV	$\mathbf{0.9375 \pm 0.0438}$	$\mathbf{0.9369 \pm 0.0460}$

Çizelge 4.6. Referans vektörlerinin ilk ağırlıklarının sınıf merkezleri olarak seçildiği ve monoton azalan öğrenme oranı ile 10-katlı çapraz doğrulama yöntemine göre optik rakam tanıma veri seti üzerinde performans analizi

Öğrenme Oranının Başlangıç Değerleri	**Yöntem**	**Ortalama Doğruluk**	**Ortalama F-ölçüm**
0.25	LVQ1	**0.9837 ± 0.0103**	**0.9181 ± 0.0527**
	LVQ2	0.9832 ± 0.0103	0.9162 ± 0.0504
	LVQ2.1	0.9832 ± 0.0103	0.9162 ± 0.0504
	LVQ3	0.9834 ± 0.0103	0.9173 ± 0.0508
	OLVQ1	0.9811 ± 0.0114	0.9060 ± 0.0551
	OLVQ3	0.9807 ± 0.0118	0.9042 ± 0.0566
	GLVQ	0.9801 ± 0.0122	0.9011 ± 0.0584
	RLVQ	0.8869 ± 0.1123	0.4045 ± 0.2595
	GRLVQ	0.9624 ± 0.0194	0.8102 ± 0.0998
	LVQRKV	**0.9813 ± 0.0116**	**0.9070 ± 0.0565**
0.5	LVQ1	**0.9837 ± 0.0103**	**0.9181 ± 0.0527**
	LVQ2	0.9832 ± 0.0103	0.9162 ± 0.0504
	LVQ2.1	0.9832 ± 0.0103	0.9162 ± 0.0504
	LVQ3	0.9834 ± 0.0103	0.9173 ± 0.0508
	OLVQ1	0.9811 ± 0.0114	0.9060 ± 0.0551
	OLVQ3	0.9807 ± 0.0118	0.9042 ± 0.0566
	GLVQ	0.9801 ± 0.0122	0.9011 ± 0.0584
	RLVQ	0.8869 ± 0.0492	0.4045 ± 0.2444
	GRLVQ	0.9813 ± 0.0116	0.9070 ± 0.0565
	LVQRKV	**0.9813 ± 0.0116**	**0.9070 ± 0.0565**
0.75	LVQ1	**0.9837 ± 0.0103**	**0.9181 ± 0.0527**
	LVQ2	0.9832 ± 0.0103	0.9162 ± 0.0504
	LVQ2.1	0.9832 ± 0.0103	0.9162 ± 0.0504
	LVQ3	0.9834 ± 0.0103	0.9173 ± 0.0508
	OLVQ1	0.9811 ± 0.0114	0.9060 ± 0.0551
	OLVQ3	0.9807 ± 0.0118	0.9042 ± 0.0566
	GLVQ	0.9801 ± 0.0122	0.9011 ± 0.0584
	RLVQ	0.8869 ± 0.0492	0.4045 ± 0.2444
	GRLVQ	0.9624 ± 0.0194	0.8102 ± 0.0998
	LVQRKV	**0.9813 ± 0.0116**	**0.9070 ± 0.0565**

Çizelge 4.7. Referans vektörlerinin ilk ağırlıklarının rastgele seçildiği ve monoton azalan öğrenme oranı ile 10-katlı çapraz doğrulama yöntemine göre optik rakam tanıma veri seti üzerinde performans analizi

Öğrenme Oranının Başlangıç Değerleri	**Yöntem**	**Ortalama Doğruluk**	**Ortalama F-ölçüm**
0.25	LVQ1	0.8202 ± 0.2387	0.0201 ± 0.0554
	LVQ2	0.8200 ± 0.2412	0.0182 ± 0.0548
	LVQ2.1	0.8200 ± 0.2412	0.0182 ± 0.0548
	LVQ3	0.8200 ± 0.2391	0.0186 ± 0.0551
	OLVQ1	0.8197 ± 0.2406	0.0180 ± 0.0543
	OLVQ3	0.8200 ± 0.2412	0.0182 ± 0.0548
	GLVQ	0.8240 ± 0.0999	0.0753 ± 0.1045
	RLVQ	0.8229 ± 0.2139	0.0417 ± 0.0799
	GRLVQ	0.8203 ± 0.2354	0.0212 ± 0.0574
	LVQRKV	$\mathbf{0.8289 \pm 0.1973}$	$\mathbf{0.0521 \pm 0.1246}$
0.5	LVQ1	0.8204 ± 0.2381	0.0219 ± 0.0612
	LVQ2	0.8200 ± 0.2412	0.0182 ± 0.0548
	LVQ2.1	0.8200 ± 0.2412	0.0182 ± 0.0548
	LVQ3	0.8201 ± 0.2387	0.0197 ± 0.0554
	OLVQ1	0.8200 ± 0.2409	0.0186 ± 0.0549
	OLVQ3	0.8200 ± 0.2411	0.0182 ± 0.0548
	GLVQ	0.8246 ± 0.1029	0.0889 ± 0.1183
	RLVQ	0.8212 ± 0.2184	0.0296 ± 0.0607
	GRLVQ	0.8207 ± 0.2360	0.0251 ± 0.0611
	LVQRKV	$\mathbf{0.8355 \pm 0.1598}$	$\mathbf{0.0647 \pm 0.1343}$
0.75	LVQ1	0.8206 ± 0.2352	0.0236 ± 0.0639
	LVQ2	0.8201 ± 0.2408	0.0196 ± 0.0561
	LVQ2.1	0.8200 ± 0.2410	0.0182 ± 0.0548
	LVQ3	0.8200 ± 0.2389	0.0187 ± 0.0552
	OLVQ1	0.8201 ± 0.2397	0.0196 ± 0.0552
	OLVQ3	0.8201 ± 0.2402	0.0189 ± 0.0550
	GLVQ	0.8175 ± 0.0960	0.0688 ± 0.1005
	RLVQ	0.8213 ± 0.2134	0.0307 ± 0.0685
	GRLVQ	0.8202 ± 0.2375	0.0208 ± 0.0554
	LVQRKV	$\mathbf{0.8348 \pm 0.1657}$	$\mathbf{0.0645 \pm 0.1429}$

Çizelge 4.8. Referans vektörlerinin ilk ağırlıklarının sınıf merkezleri olarak seçildiği ve monoton azalan öğrenme oranı ile 10-katlı çapraz doğrulama yöntemine göre doküman sınıflandırma veri seti üzerinde performans analizi

Öğrenme Oranının Başlangıç Değerleri	**Yöntem**	**Ortalama Doğruluk**	**Ortalama F-ölçüm**
0.25	LVQ1	0.8958 ± 0.0687	0.7873 ± 0.1434
	LVQ2	0.9021 ± 0.0623	0.8024 ± 0.1283
	LVQ2.1	0.9021 ± 0.0623	0.8024 ± 0.1283
	LVQ3	$\mathbf{0.9104 \pm 0.0684}$	$\mathbf{0.8147 \pm 0.1565}$
	OLVQ1	0.8271 ± 0.0815	0.6280 ± 0.2221
	OLVQ3	0.8875 ± 0.0855	0.7716 ± 0.1733
	GLVQ	0.8208 ± 0.0977	0.6227 ± 0.2384
	RLVQ	0.8438 ± 0.1028	0.6732 ± 0.2209
	GRLVQ	0.8333 ± 0.1016	0.6645 ± 0.1924
	LVQRKV	$\mathbf{0.8625 \pm 0.0876}$	$\mathbf{0.7053 \pm 0.2177}$
0.5	LVQ1	$\mathbf{0.8896 \pm 0.0722}$	$\mathbf{0.7731 \pm 0.1558}$
	LVQ2	0.8688 ± 0.0798	0.7390 ± 0.1530
	LVQ2.1	0.8688 ± 0.0798	0.7390 ± 0.1530
	LVQ3	0.8833 ± 0.0802	0.7645 ± 0.1641
	OLVQ1	0.7917 ± 0.1266	0.5440 ± 0.2807
	OLVQ3	0.8938 ± 0.0687	0.7843 ± 0.1420
	GLVQ	0.8208 ± 0.0977	0.6227 ± 0.2384
	RLVQ	0.8458 ± 0.0986	0.6870 ± 0.1876
	GRLVQ	0.8104 ± 0.1140	0.6127 ± 0.2331
	LVQRKV	$\mathbf{0.8792 \pm 0.0771}$	$\mathbf{0.7511 \pm 0.1717}$
0.75	LVQ1	$\mathbf{0.8833 \pm 0.0824}$	$\mathbf{0.7652 \pm 0.1513}$
	LVQ2	0.8500 ± 0.0914	0.6938 ± 0.1864
	LVQ2.1	0.8500 ± 0.0914	0.6938 ± 0.1864
	LVQ3	0.8458 ± 0.0935	0.6808 ± 0.2204
	OLVQ1	0.8167 ± 0.0943	0.6081 ± 0.2449
	OLVQ3	0.8813 ± 0.0787	0.7587 ± 0.1618
	GLVQ	0.8208 ± 0.0977	0.6227 ± 0.2384
	RLVQ	0.8271 ± 0.1057	0.6373 ± 0.2265
	GRLVQ	0.8000 ± 0.0945	0.6009 ± 0.1715
	LVQRKV	$\mathbf{0.8646 \pm 0.0788}$	$\mathbf{0.7203 \pm 0.1772}$

Çizelge 4.9. Referans vektörlerinin ilk ağırlıklarının rastgele seçildiği ve monoton azalan öğrenme oranı ile 10-katlı çapraz doğrulama yöntemine göre doküman sınıflandırma veri seti üzerinde performans analizi

Öğrenme Oranının Başlangıç Değerleri	**Yöntem**	**Ortalama Doğruluk**	**Ortalama F-ölçüm**
0.25	LVQ1	0.7444 ± 0.1856	0.1427 ± 0.2041
	LVQ2	0.7324 ± 0.2135	0.0834 ± 0.1426
	LVQ2.1	0.7287 ± 0.2218	0.0714 ± 0.1329
	LVQ3	0.7731 ± 0.1429	0.2415 ± 0.2288
	OLVQ1	0.7398 ± 0.2007	0.1240 ± 0.2225
	OLVQ3	0.8009 ± 0.1528	0.3502 ± 0.2755
	GLVQ	0.7185 ± 0.1676	0.1083 ± 0.1537
	RLVQ	0.7509 ± 0.2152	0.1541 ± 0.2570
	GRLVQ	0.7602 ± 0.1303	0.1938 ± 0.2241
	LVQRKV	$\mathbf{0.8176 \pm 0.0733}$	$\mathbf{0.4114 \pm 0.2552}$
0.5	LVQ1	0.7343 ± 0.1842	0.1162 ± 0.1804
	LVQ2	0.7333 ± 0.2189	0.0973 ± 0.1549
	LVQ2.1	0.7296 ± 0.2280	0.0883 ± 0.1463
	LVQ3	0.7657 ± 0.1538	0.2154 ± 0.2651
	OLVQ1	0.7343 ± 0.2378	0.1000 ± 0.1913
	OLVQ3	0.7981 ± 0.1330	0.3440 ± 0.2163
	GLVQ	0.7148 ± 0.1433	0.1060 ± 0.1356
	RLVQ	0.7222 ± 0.2505	0.0476 ± 0.1074
	GRLVQ	0.7491 ± 0.1471	0.1685 ± 0.2228
	LVQRKV	$\mathbf{0.8120 \pm 0.0889}$	$\mathbf{0.4111 \pm 0.2483}$
0.75	LVQ1	0.7352 ± 0.2081	0.1058 ± 0.1626
	LVQ2	0.7343 ± 0.2036	0.0989 ± 0.1680
	LVQ2.1	0.7324 ± 0.2081	0.0952 ± 0.1743
	LVQ3	0.7500 ± 0.1651	0.1628 ± 0.1996
	OLVQ1	0.7287 ± 0.1783	0.0985 ± 0.1806
	OLVQ3	0.8167 ± 0.1154	0.4288 ± 0.2310
	GLVQ	0.7074 ± 0.1454	0.0932 ± 0.1200
	RLVQ	0.7324 ± 0.2430	0.0861 ± 0.1819
	GRLVQ	0.7491 ± 0.1452	0.1752 ± 0.2005
	LVQRKV	$\mathbf{0.8185 \pm 0.0776}$	$\mathbf{0.4159 \pm 0.2524}$

5. BULGULAR VE TARTIŞMA

5.1. Sonuçlar

Tez çalışmasında, Kohonen tarafından önerilen genelleştirilmiş delta öğrenme kuralı kullanılırken karşılaşılan problemi ortadan kaldırmak için yeni bir öğrenme algoritması önerilmiştir. Bu yöntem iris, meme kanseri tespiti, optik karakter tanıma, doküman sınıflandırma olmak üzere, dört farklı veri seti üzerinde uygulanmış ve performansı LVQ ağlarının diğer çeşitleriyle karşılaştırılmıştır. Önerilen yaklaşım mevcut iş yükünü ve keyfi seçilen parametrelere olan bağımlılığı azaltarak diğer vektör nicemleme yaklaşımlarının gösterdiği başarıyı elde etmiştir.

Çok sayıda yapılan deneysel çalışmalardan elde edilen sonuçlar, önerilen öğrenme algoritmasının performansını doğrulamaktadır. Buna ek olarak, genelleştirilmiş delta öğrenme kuralı kullanılırken karşılaşılan durum, LVQRKV metodunu kullanırken çok düşük olasılıkla gerçekleşebilir. Bu durumun gözlemlenebilmesi için aynı örneğin arka arkaya gösterilmesi gerekmektedir. Şekil 3.8, Şekil 3.9 ve Şekil 3.10'da açıklanan üç durum, aynı örnek ağa arka arkaya gösterilmediğinden çalışmamızda hiç gerçekleşmemiştir.

Genel olarak elde edilen sonuçlar, önerilen metodun genelleştirilmiş delta öğrenme kuralını geliştirmek için önemli ve dikkat çekici bir öğrenme algoritması olduğunu göstermiştir. Önerilen öğrenme algoritmasının 'kazanan her şeyi alır' prensibine göre çalışan diğer algoritmalar için kullanılabilir olduğu fikri ortaya çıkmıştır.

KAYNAKLAR

[1] Alpaydin, E. 2013. Yapay Öğrenme Nedir? Yapay Öğrenme, Boğaziçi Üniversitesi Yayınevi, pp. 1-34, Türkiye.

[2] Simon, P. 2013. Too Big to Ignore: The Business Case for Big Data. In: John Wiley and Sons, pp. 89, Canada.

[3] Mitchell, T. 1997. Machine Learning. In: McGraw-Hill Series in Computer Sciences, Carnegie Mellon University, pp. 2, The United States of America.

[4] Epstein, R.,Roberts, G. and Beber, G. 2009. Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer. Springer, pp. 511, The United States of America.

[5] Barber, D. 2012. Machine Learning. In:Bayesian Reasoning and Machine Learning, Cambridge University Press, pp. 285-336, New York.

[6] Watanabe, S. 2009. Algebraic Geometry and Statistical Learning Theory (Cambridge Monographs on Applied and Computational Mathematics), Cambridge University Press, The United States of America.

[7] John, G. 1993. **Geometry-Based Learning Algorithms** (Unpublished).

[8] Kim, J. H. and Park, S. 1995. The geometrical learning of binary neural networks. **IEEE Transactions on Neural Networks**, 6 (1) : 237-247.

[9] Cabrelli, C., Molter, U. and Shonkwiler, R. 2000. A constructive algorithm to solve “convex recursive deletion" (CoRD) classification problems via two-layer perceptron networks. **IEEE Transactions on Neural Networks and Learning Systems**, 11 (3): 811-816.

[10] Wang, D. and Chaudhari, N. S. 2004. An approach for construction of Boolean neural networks based on geometrical expansion. **Neurocomputing**, 57: 455-461.

[11] Shoujue, W. and Jiangliang, L. 2005. Geometrical learning, descriptive geometry, and biomimetic pattern recognition. **Neurocomputing**, 67: 9-28.

[12] Bayro-Corrochano E. and Anana-Daniel N. 2005. MIMO SVMs for classification and regression using the geometric algebra framework. In **Proceedings of International Joint Conference on Neural Networks**, pp. 895-900.

[13] Zhang D., Chan X. and Lee W. S. 2005. Text classification with kernels on the multinomial manifold. In **In SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval**, pp. 266-273.

[14] Delogu, R., Fanni, A. and Montisci, A. 2008. Geometrical synthesis of MLP neural networks. **Neurocomputing**, 71 (4-6) : 919-930.

[15] Liu, Z., Liu, J. G., Pan, C. and Wang, G. 2009. A novel geometric approach to binary classification based on scaled convex hulls. **IEEE Transactions on Neural Networks**, 20 (7) : 1215-1220.

[16] Wang, D., Qiao, H., Zhang, B. and Wang, M. 2013. Online support vector machine based on convex hull vertices selection. **IEEE Transactions on Neural Networks and Learning Systems**, 24 (4) : 593-609.

[17] Kohonen, T. 1990. The self-organizing map. In **Proceedings of the IEEE**, 78 (9), pp. 1464-1480.

[18] Kohonen, T. 1990. Improved versions of learning vector quantization. **In Proceedings of the International Joint Conference on Neural Networks**, 1, pp. 545-550, San Diego.

[19] Sato, A. S. and Yamada, K. (1995). Generalized learning vector quantization. **Advances in Neural Information Processing Systems**, 7: 423-429, MIT Press.

[20] Kohonen, T., Hynninen, J., Kangas, J., Laaksonen, J. and Torkkola, K. 1996. LVQ_PAK: The learning vector quantization program package. Helsinki University of Technology, **Laboratory of Computer and Information Science**, Finland.

[21] Bojer. T., Hammer, B., Schunk D. and Tluk von Toschanowitz K. 2001. Relevance determination in learning vector quantization. **In Proc. of European Symposium on Artificial Neural Networks**, pp. 271 - 276, Brussels, Belgium.

[22] Hammer, B. and Villmann, T. 2002. Generalized relevance learning vector quantization. **Neural Networks - New developments in self-organizing maps**, 15 (8-9) : 1059-1068.

[23] Seo, S., Obermayer, K 2003. Soft learning vector quantization. **Neural Computation**, 15(7) : 1589-1604.

[24] Schneider, P., Hammer, B. and Biehl. M. 2009. Adaptive relevance matrices in learning vector quantization. **Neural Computation**, 21: 3532-3561.

[25] Kaestner, M., Hammer, B., Biehl, M., Villmann, T. 2012. Functional relevance learning in generalized learning vector quantization. **Neurocomputing**, 90: 85-95.

[26] Biehl M., Ghosh A., Hammer B. and Bengio Y. 2007. Dynamics and Generalization Ability of LVQ Algorithms, **The Journal of Machine Learning Research**, 8: 323-360.

[27] Bache, K., Lichman, M. 2013. UCI Repository of Machine Learning Databases, Irvine, CA: University of California, School of Information and Computer Science, [`http://archive.ics.uci.edu/ml/`], Erişim Tarihi:01.05.2014.

[28] Wall, M. E., Rechtsteiner, A. and Rocha, L. M. 2003. Singular value decomposition and principal component analysis. **A Practical Approach to Microarray Data Analysis**. 91-109.

[29] Fisher, Ronald A. 1936. The use of multiple measurements in taxonomic problems. **Annals of Eugenics**, 2: 179-188.

[30] Wolberg, W.H., Street, W.N., Heisey, D.M., Mangasarian, O.L. 1995. Computer-derived nuclear features distinguish malignant from benign breast cytology. **Human Pathology**, 26: 792-796.

[31] Kaynak, C. 1995. Methods of Combining Multiple Classifiers and Their Applications to Handwritten Digit Recognition. Bogazici University, Institute of Graduate Studies in Science and Engineering, MSc Thesis, Istanbul.

[32] Alpaydin, E., Kaynak, C. 1998. Cascaded Classifiers. **Kybernetika**, 34: 369-374.

EKLER

Verilen kodlar, ağırlıkların ilk değerlerinin rastgele seçilmesiyle oluşturulan ağ topolojisinin iris veriseti üzerinde çalışması amacıyla hazırlanmıştır.

Ek 1. $\mathbb{R}^n$ verilen üç noktanın doğrusal olup olmadığını belirleyen fonksiyon

```
function y=dogrusalmi(A,B,C)
  u1=(B-A)/(pdist2(A,B,'euclidean'));
  u2=(C-B)/(pdist2(C,B,'euclidean'));
  y=sum(u1==u2)==size(u1,2);
end % function
```

Ek 2. Verilen noktanın hiper-kürenin dışında olup olmadığını belirleyen fonksiyon

```
function y=noktaHiperKureninDisindami(M,r,P)
% n boyutlu uzayda P noktasının M merkez ve r yarıçaplı
% kürenin dışında olup olmadığını inceler.
% P hiper-kürenin dış bölgesinde ise fonksiyon sonucu 1'dir.
% P hiper-kürenin iç bölgesinde ise fonksiyon sonucu -1'dir.
% P hiper-kürenin üzerindeyse fonksiyonun sonucu 0'dir.

  t=sum((P-M).^2);
  if (t<r^2) % P noktası hiperkürenin iç bölgesinde
    y=-1;
  else
    if (t==r^2) % P noktası hiperkürenin iç bölgesinde
      y=0;
    else
      y=1;
    end
  end
end %function
```

Ek 3. Verilen noktanın ötelenip hiper-küre üzerinde döndürülmesini sağlayan fonksiyon

```
function y = hiperkureUzerindeOteleme(M,r,dphi,w,yon)
% n-boyutlu uzayda M merkezli r yarıçaplı hiperküre üzerinde
% yer alan P noktasını hiperküreyi ilk eksen ektarında
% dphi açısı kadar döndürülünce yeni noktanın
% koordinatlarını veren fonksiyon
% M - Hiperkürenin merkez koordinatları
% r - Hiperkürenin yarıçapı
% dphi - Döndürme açısı
% w - Hiperküre üzerindeki bir nokta

  w=w-M; % Nokta orjine taşınıyor
  n = size(M,2); % Uzayın boyutu

  % w noktası hiper küre üzerinde olduğu için denklemi sağlar.
  % O halde açı değerlerini hesaplatabiliriz.
  phi=zeros(1,n-1);
  for i=1:n-2
    toplam=0;
    for j=i+1:n
        toplam=toplam + w(j)^2;
    end
    phi(i)=acot(w(i)/sqrt(toplam));
  end
  phi(n-1)=2*acot((sqrt(w(n-1)^2+w(n)^2)+w(n-1))/w(n));

  % w noktasını ilk eksen yönünde dphi kadar döndürürsek
  % yeni koordinatlar ne olur?
  phi(1)=phi(1)-yon*(pi+dphi);
  w(1)=r*cos(phi(1));
  c=1;
  for i=2:n-1
    c=1;
    for k=1:i-1
        c = c*sin(phi(k));
    end
    w(i) = r*c*cos(phi(i));
  end
  w(n) =r*c*sin(phi(n-1));

  % Nokta döndürme işlemi tamamlandıktan sonra tekrar öteleniyor
  w=w+M;
  y=w;
end % function
```

Ek 4. k-katlı çarpraz doğrulama için örneklem kümesini k eş parçaya ayıran fonksiyon

```
function [kume,egitim,test]=cross_validation_partition(girdi,cikti,k)
% Bu fonksiyon bir veri setini k eş parçaya böler.
% Bölünme sonunda k adet kumede her sınıftan eşit sayıda örnek bulunur.
% girdi - her satır bir girdi vektörünü temsil eder
% cikti -satır vektörüdür
% k - Veri setinin bölüneceği küme sayısı

  fprintf('Veri seti %d eşit parçaya bölünüyor...',k);
  %% Her bir sınıftaki örneklerin indislerini bul
  % cikti vektörü içindeki farklı sınıf isimlerini bul
  siniflar = unique(cikti);
  ns = size(siniflar,2); % sınıf sayısı
  sinifa_ait_ornek_indisleri=zeros(size(siniflar,2), size(cikti,2));

  for i=1:size(siniflar,2)
    indisler = findstr(cikti, siniflar(i));
    for j=1:size(indisler,2)
       sinifa_ait_ornek_indisleri(i,j)=indisler(j);
    end
  end

  % Her bir sınıftaki örnek sayıları
  for i=1:size(siniflar,2)
     sinif_ornek_sayisi(i)=nnz(sinifa_ait_ornek_indisleri(i,:));
         % Sıfırdan farklı eleman sayısını bulur
  end

  fark =mod(min(sinif_ornek_sayisi), lcm(ns,k));
  m =min(sinif_ornek_sayisi) -fark;
  sayac=1;
  for kume_no=1:k
    kume(kume_no).P=[];
    kume(kume_no).T=[];
    j=1;
    while (j≤m*ns/k)
      ind=rem(j,ns);  % Eklenecek örneğin indisi
      if ind==0
        ind=siniflar(size(siniflar,2));
      end
      x=girdi(sinifa_ait_ornek_indisleri(ind,sayac),:);
      kume(kume_no).P= [kume(kume_no).P; x];
      kume(kume_no).T=[kume(kume_no).T ...
            cikti(sinifa_ait_ornek_indisleri(ind,sayac))];
      if (rem(j,ns)==0) sayac=sayac+1; end
      j=j+1;
    end
  end

  % Her bir küme içindeki örnekleri sınıflarına göre sırala
```

```
50    for kume_no=1:k
51      % Sıralama
52      for i=1:size(kume(kume_no).T,2)-1
53        for j=i+1:size(kume(kume_no).T,2)
54          if kume(kume_no).T(i)>kume(kume_no).T(j)
55            geciciP=kume(kume_no).P(i,:);
56            kume(kume_no).P(i,:)=kume(kume_no).P(j,:);
57            kume(kume_no).P(j,:)=geciciP;
58
59            geciciT=kume(kume_no).T(i);
60            kume(kume_no).T(i)=kume(kume_no).T(j);
61            kume(kume_no).T(j)=geciciT;
62          end
63        end
64      end
65    end
66    % Eğitim ve Test kümeleri oluşturuluyor
67    sayac=1;
68    for i=1:size(kume,2)
69      % i. küme test, geri kalan kümelerin birleşimi
70      % eğitim için kullanılacak
71      test(sayac).P=kume(i).P;
72      test(sayac).T=kume(i).T;
73      egitim(sayac).P=[];
74      egitim(sayac).T=[];
75
76      for j=1:size(kume,2)
77        if (i~=j)
78          egitim(sayac).P=[ egitim(sayac).P; kume(j).P];
79          egitim(sayac).T=[ egitim(sayac).T kume(j).T];
80        end
81      end
82      % egitim(sayac) kümesindeki örnekleri sınıflara göre sırala
83      for ii=1:size(egitim(sayac).T,2)-1
84        for jj=ii+1:size(egitim(sayac).T,2)
85          if egitim(sayac).T(ii)>egitim(sayac).T(jj)
86            geciciP=egitim(sayac).P(ii,:);
87            egitim(sayac).P(ii,:)=egitim(sayac).P(jj,:);
88            egitim(sayac).P(jj,:)=geciciP;
89
90            geciciT=egitim(sayac).T(ii);
91            egitim(sayac).T(ii)=egitim(sayac).T(jj);
92                   egitim(sayac).T(jj)=geciciT;
93          end
94        end
95      end
96      sayac=sayac+1;
97    end
98  end % function
```

Ek 5. LVQRKV metodu için parametre değerlerinin belirlendiği betik

```
clear all
clc
load iris_dataset;
P=irisInputs'; % Örneklem uzayı
for i=1:size(irisTargets,2)
    T(i)= findstr(irisTargets (:,i)',1); % Çıktı değerleri
end
clear irisInputs irisTargets;

% Örneklem uzayı 10 kümeye ayrılıyor
k=10;
[kume,egitim,test]=cross_validation_partition(P,T,k);

siniflar = unique(T);
referans_vektor_sayisi =10;
topoloji = referans_vektor_sayisi*ones(1,size(siniflar,2));
Ogrenme_Orani = 0.75;
Monoton_Azalma_Yuzdesi = 0.001;
Hedef = 1; % Eğitimin biteceği performansıdır.
Maksimum_Iterasyon = 300;

save parametreler.mat
```

Ek 6. LVQ_RKV metodu için eğitim ve test işlemlerini gerçekleştiren betik

```
for kume_no=1:k
  P=egitim(kume_no).P;
  % Her satır bir girdi vektörünü temsil eder.
  T=egitim(kume_no).T;

  % İlk ağırlıklar
  a=-1;
  b=1;
  for i=1:sum(topoloji)
    rng('shuffle');
    Agirliklar{i}= a + (b-(a)).*rand(size(P,2),1);
  end
  disp('LVQ_RKV');

  % Eğitim
  NET_LVQ_RKV=LVQ_RKV(P',T,topoloji,Ogrenme_Orani,...
    Monoton_Azalma_Yuzdesi,Hedef,Maksimum_Iterasyon,Agirliklar);

  % Test Aşaması başlıyor.
  P_test =  test(kume_no).P';
  % Her sütun bir girdi vektörünü temsil eder.
  T_test = test(kume_no).T;

  % sinifBasSon değerleri belirleniyor.
  sinifBasSon=zeros(size(siniflar,2),2);
  for i=1:size(siniflar,2)
    clear indis;
    indis=findstr(T_test,siniflar(i));
    sinifBasSon(i,1)=indis(1);
    sinifBasSon(i,2)=indis(size(indis,2));
  end

  ToplamSonuc=[];
  for sinif=1:size(sinifBasSon,1)
    % sınıf sayısı
    clear sonuc;
    sonuc=simulation(NET_LVQ_RKV,P_test...
      (:,sinifBasSon(sinif,1):sinifBasSon(sinif,2)));
    ToplamSonuc=[ToplamSonuc;sonuc];
  end

  for i=1:size(siniflar,2)
    sinif_isimleri(i)={num2str(siniflar(i))};
  end
  [TumSiniflar_Accuracy_Ortalamasi,...
    TumSiniflar_Fmeasure_Ortalamasi,TP,FN,...
    FP,TN,Recall,Precision,Fmeasure,Accuracy]=...
    Evaluate(ToplamSonuc,sinif_isimleri);
  sonuclar.Deney(kume_no).ToplamOrnekSayisi=...
    size(T_test,2);
```

```
51    sonuclar.Deney(kume_no).HataliTespitEdilenOrnekSayisi=...
52      sum(FN)+sum(FP);
53    sonuclar.Deney(kume_no).TumSiniflar_Accuracy_Ortalamasi=...
54      TumSiniflar_Accuracy_Ortalamasi;
55    sonuclar.Deney(kume_no).TumSiniflar_Fmeasure_Ortalamasi=...
56      TumSiniflar_Fmeasure_Ortalamasi;
57    sonuclar.Deney(kume_no).TP=TP;
58    sonuclar.Deney(kume_no).FN=FN;
59    sonuclar.Deney(kume_no).FP=FP;
60    sonuclar.Deney(kume_no).TN=TN;
61    sonuclar.Deney(kume_no).Recall=Recall;
62    sonuclar.Deney(kume_no).Precision=Precision;
63    sonuclar.Deney(kume_no).Fmeasure=Fmeasure;
64    sonuclar.Deney(kume_no).Accuracy=Accuracy;
65  end % end of for kume_no
66  TumDeneyOrnekleriToplami=k*size(T_test,2);
67
68  save('sonuclar_LVQ_RKV.mat','sonuclar',...
69    'siniflar','k','TumDeneyOrnekleriToplami');
```

Ek 7. LVQ_RKV metodunu gerçekleştiren fonksiyon

```
function NET=LVQ_RKV(P,T,topoloji,Ogrenme_Orani,...
  Monoton_Azalma_Yuzdesi,Hedef,Maksimum_Iterasyon,Agirliklar)
  format long
  dogrusal_nokta_say=0;

  % İlk değerler atanıyor
  NET.P=P;
  NET.T=T;
  NET.LVQ_Topoloji= topoloji;
  NET.Ogrenme_Orani=Ogrenme_Orani;
  NET.monoton_azalma_yuzdesi=Monoton_Azalma_Yuzdesi;
  NET.hedef=Hedef;  % Eğitimin biteceği performansıdır
  max_iter=Maksimum_Iterasyon;
  NET.performans=0; % Başlangıçta 0 olarak kabul edilir
  NET.toplam_proseseleman=sum(NET.LVQ_Topoloji);
  epsilon=0.2;

  % Her sınıfın merkez vektörleri hesaplanır
  for k=1:length(NET.LVQ_Topoloji)
    M{k}=[];
  end
  for t=1:size(NET.T,2)
    M{NET.T(t)}=[M{NET.T(t)} NET.P(:,t)];
  end
  for k=1:length(NET.LVQ_Topoloji)
    Center{k}=(sum(M{k}')./size(M{k},2))';
  end

  NET.Kohonen_Vektorleri = Agirliklar; % İlk ağırlık değerleri
  window_width=0.2;
  s2=(1-window_width)/(1+window_width);

  % Eğitim işlemi başlıyor
  iterasyon=0;
  while (NET.performans<NET.hedef)
    for i=1:length(NET.T) % Girdi vektörlerinin sayısı
      iterasyon=iterasyon+1;
      if ((iterasyon>max_iter) || (NET.performans==NET.hedef))
        fprintf('%d. iterasyonda hedefe ulaşıldı...\n',iterasyon-1);
        NET_LVQ_RKV=NET;
        save NET_LVQ_RKV.mat NET_LVQ_RKV;
        simulation(NET,NET.P)
        return
      end
      input=NET.P(:,i);

      % Global Kazanan vektör hesaplanır öklit uzaklığına göre
      for j=1:NET.toplam_proseseleman
        uzakliklar(j)=sum((input-NET.Kohonen_Vektorleri{j}).^2).^0.5;
      end
```

```
51      [K1 GlobalKazananProsesElemani]=min(uzakliklar);
52      % Global kazananın hangi sınıfta olduğu bulunacak
53      ToplamProsesElemani=0;
54      for sinif=1:size(NET.LVQ_Topoloji,2)
55        ToplamProsesElemani=ToplamProsesElemani+...
56                  NET.LVQ_Topoloji(sinif);
57          if (GlobalKazananProsesElemani<=...
58                        ToplamProsesElemani)
59              GlobalKazananProsesElemanininSinifi = sinif;
60              break
61          end
62      end
63      % Her sınıfa göre vektörlerin başlangıç
64          % ve bitiş indisleri bulunur.
65      Toplam=0;
66      for sinif=1:length(NET.LVQ_Topoloji)
67        SinifVektorBas(sinif)=Toplam+1;
68        Toplam=Toplam+NET.LVQ_Topoloji(sinif);
69        SinifVektorSon(sinif)=Toplam;
70      end
71
72      yereluzakliklar(1:NET.toplam_proseseleman)=inf;
73      for uz=SinifVektorBas(NET.T(i)):...
74            SinifVektorSon(NET.T(i))
75        yereluzakliklar(uz)=sum((input-...
76                        NET.Kohonen_Vektorleri{uz}).^2).^0.5;
77      end
78      [K2 YerelKazananProsesElemani]=min(yereluzakliklar);
79
80      % Yerel kazanan proses elemanının hangi
81      % sınıfta olduğu bulunacak
82      ToplamProsesElemani=0;
83      for sinif=1:size(NET.LVQ_Topoloji,2)
84        ToplamProsesElemani=ToplamProsesElemani+...
85               NET.LVQ_Topoloji(sinif);
86        if (YerelKazananProsesElemani<=ToplamProsesElemani)
87           YerelKazananProsesElemanininSinifi = sinif;
88           break
89        end
90      end
91      % Ağırlıklar güncelleniyor
92      P=input';
93      s=-1;
94      M_A=Center{YerelKazananProsesElemanininSinifi}';
95      M_B=Center{GlobalKazananProsesElemanininSinifi}';
96
97      % Yerel Kazanan ve Global Kazanan arasındaki
98      % uzaklığı çap kabul eden hiperküre - tanımlanıyor.
99      yaricap = norm(M_A-M_B);
100     merkez = (M_A+M_B)/2;
101     if (noktaHiperKureninDisindami(P,yaricap,merkez)~=1)...
102        && (noktaHiperKureninDisindami(P,...
```

```
103           yaricap*epsilon,merkez)==1)
104            if (GlobalKazananProsesElemanininSinifi==...
105             YerelKazananProsesElemanininSinifi)
106             % Bu durumda aynı sınıftadırlar, yaklaştırılacak
107             if (dogrusalmi(P,X,M_A)==1)
108             % Kohonen vektörü X sadece girdi vektörü
109                     % P'ye yaklaştırılacak
110               NET.Kohonen_Vektorleri{GlobalKazananProsesElemani}=...
111                 NET.Kohonen_Vektorleri{GlobalKazananProsesElemani}...
112                  +NET.Ogrenme_Orani*(input-...
113                  NET.Kohonen_Vektorleri{GlobalKazananProsesElemani});
114                 dogrusal_nokta_say=dogrusal_nokta_say+1;
115               else
116                 % Kohonen vektörü döndürülerek hem P'ye
117                 % hem de M_A'ya yaklaştırılacak
118                 r1=pdist2(M_A,X,'euclidean')/2;
119                 r2 = pdist2(P,X,'euclidean');
120                 C=(M_A+X)/2;
121
122                 % Döndürme açısı hesaplanıyor.
123                 alpha =X(1)/norm(X,2);
124                 beta1=alpha;
125                 beta2=max(acos(C(1)/norm(C,2)),acos(P(1)/norm(P,2)));
126                 alpha=alpha-s*NET.Ogrenme_Orani*abs(beta2-beta1);
127
128                 % P ve M_A noktasına yakınlaşacak
129                 x2=hiperkureUzerindeOteleme(C,r1,alpha,X,1);
130                 x3=hiperkureUzerindeOteleme(C,r1,alpha,X,-1);
131
132                 if (noktaHiperKureninDisindami(P,r2,x2)==1)
133                   if (noktaHiperKureninDisindami(P,r2,x3)==1)
134                     NET.Kohonen_Vektorleri{YerelKazananProsesElemani}...
135                      =NET.Kohonen_Vektorleri{YerelKazananProsesElemani}...
136                      +NET.Ogrenme_Orani*(input...
137                      -NET.Kohonen_Vektorleri{YerelKazananProsesElemani});
138                   else
139                     NET.Kohonen_Vektorleri{YerelKazananProsesElemani}...
140                      =x3';
141                   end
142                 else
143                   NET.Kohonen_Vektorleri{YerelKazananProsesElemani}...
144                    =x2';
145                 end
146               end % if dogrusalmi
147            else
148             if (dogrusalmi(P,X,M_A)==1)
149               NET.Kohonen_Vektorleri{GlobalKazananProsesElemani}=...
150               NET.Kohonen_Vektorleri{GlobalKazananProsesElemani}+...
151               NET.Ogrenme_Orani*(input-NET.Kohonen_Vektorleri...
152                {GlobalKazananProsesElemani});
153                dogrusal_nokta_say=dogrusal_nokta_say+1;
154                                       else
```

```
155                 r1=pdist2(M_A,X,'euclidean')/2;
156                 r2 = pdist2(P,X,'euclidean');
157                 C=(M_A+X)/2;
158
159                 alpha =X(1)/norm(X,2);
160                 beta1=alpha;
161                 beta2=max(acos(C(1)/norm(C,2)),acos(P(1)/norm(P,2)));
162                 alpha=alpha-s*NET.Ogrenme_Orani*abs(beta2-beta1);
163                 x2=hiperkureUzerindeOteleme(C,r1,alpha,X,1);
164                 x3=hiperkureUzerindeOteleme(C,r1,alpha,X,-1);
165
166                 if (noktaHiperKureninDisindami(P,r2,x2)==1)
167                   if (noktaHiperKureninDisindami(P,r2,x3)==1)
168                     NET.Kohonen_Vektorleri{YerelKazananProsesElemani}=...
169                       NET.Kohonen_Vektorleri{YerelKazananProsesElemani}...
170                       +NET.Ogrenme_Orani*(input-...
171                       NET.Kohonen_Vektorleri{YerelKazananProsesElemani});
172                   else
173                       NET.Kohonen_Vektorleri{YerelKazananProsesElemani}=x3';
174                   end
175                 else
176                   NET.Kohonen_Vektorleri{YerelKazananProsesElemani}=x2';
177                 end
178               end % if dogrusalmi
179
180               if (dogrusalmi(P,Y,M_B)==1)
181                 NET.Kohonen_Vektorleri{GlobalKazananProsesElemani}=...
182                   NET.Kohonen_Vektorleri{GlobalKazananProsesElemani}...
183                   -NET.Ogrenme_Orani*(input-...
184                   NET.Kohonen_Vektorleri{GlobalKazananProsesElemani});
185               else
186                 r1=pdist2(M_B,Y,'euclidean')/2;
187                 r2 = pdist2(P,Y,'euclidean');
188                 M=(M_B+Y)/2;
189                 alpha =Y(1)/norm(Y,2);
190                 beta1=alpha;
191                 beta2=max(acos(C(1)/norm(C,2)),acos(P(1)/norm(P,2)));
192                 alpha=alpha-s*NET.Ogrenme_Orani*abs(beta2-beta1);
193
194                 y2=hiperkureUzerindeOteleme(M,r1,alpha,Y,1);
195                 y3=hiperkureUzerindeOteleme(M,r1,alpha,Y,-1);
196                 if (noktaHiperKureninDisindami(P,r2,y2)==1)
197                   NET.Kohonen_Vektorleri{GlobalKazananProsesElemani}=y2';
198                 else
199                   NET.Kohonen_Vektorleri{GlobalKazananProsesElemani}=y3';
200                 end
201                 if (noktaHiperKureninDisindami(P,r2,y2)==1)
202                  if (noktaHiperKureninDisindami(P,r2,y3)==1)
203                    NET.Kohonen_Vektorleri{GlobalKazananProsesElemani}...
204                     =NET.Kohonen_Vektorleri{GlobalKazananProsesElemani}...
205                     -NET.Ogrenme_Orani*(input-...
206                     NET.Kohonen_Vektorleri{GlobalKazananProsesElemani});
```

```
207                 else
208                   NET.Kohonen_Vektorleri{GlobalKazananProsesElemani}...
209                                   =y2';
210                 end
211               else
212                 NET.Kohonen_Vektorleri{GlobalKazananProsesElemani}...
213                                   =y3';
214               end
215             end % if dogrusalmi
216           end
217       end
218
219       ciktilar=simulation(NET,NET.P);
220       t=0;
221       for ind=1:size(NET.T,2)
222         t=t+isequal(NET.T(ind),ciktilar(ind));
223       end
224       NET.performans=t/size(NET.T,2);
225
226        if rem(iterasyon,10)==0
227          fprintf('İterasyon: %d   Performans: %2.2f\n',...
228          iterasyon,NET.performans);
229        end
230        NET.Ogrenme_Orani=NET.Ogrenme_Orani-...
231          NET.monoton_azalma_yuzdesi*NET.Ogrenme_Orani;
232      end % end of for i
233    end % end of while
234
235    NET_LVQ_RKV=NET;
236    save NET_LVQ_RKV.mat NET_LVQ_RKV;
237
238    simulation(NET,NET.P)
239    disp(dogrusal_nokta_say)
240  end % function
```

ÖZ GEÇMİŞ

KİŞİSEL BİLGİLER

Adı Soyadı : İclal GÖR
Doğum Yeri ve Tarihi : İspir, 01.03.1991

EĞİTİM DURUMU

Lisans Öğrenimi : Mimar Sinan Güzel Sanatlar Üni.
Fen-Edebiyat Fak., Matematik Böl.
Yüksek Lisans Öğrenimi : Adnan Menderes Üniversitesi
Fen-Edebiyat Fak., Matematik Böl.
Bildiği Yabancı Diller : İngilizce

BİLİMSEL FAALİYETLERİ

a) Yayınlar
-SCI :

Ulvi İ., Bayrak M.A., Identifying an unknown function in a parabolic equation by homotopy analysis method and comparison with the Adomian decomposition method, AIP (American Institute of Physics) Conference Proceedings, 1470, 88-91, 2012.

-Diğer
b) Bildiriler
-Uluslararası :

Ulvi İ., Asliyan R., Gunel K., Dincel O., Cagirici A., Textile Image Classification Using Artificial Neural Networks, AWER Procedia Information Technology and Computer Science vol. 4, pp.615-620, 2013.

-Ulusal
c) Katıldığı Projeler

İŞ DENEYİMİ

Çalıştığı Kurumlar ve Yıl : Adnan Menderes Üniversitesi,
Fen-Edebiyat Fak. Matematik Böl.
(2012 - ...)

İLETİŞİM

E-posta Adresi : iclal@adu.edu.tr
Tarih :