İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

İŞARETLERİN SEYREK GÖSTERİLİMİ İÇİN SÖZLÜK TASARIMI

YÜKSEK LİSANS TEZİ

Emrah YAVUZ
(504081311)

Tezin Enstitüye Verildiği Tarih : 5 Mayıs 2011
Tezin Savunulduğu Tarih : 7 Haziran 2011

Tez Danışmanı : Yrd. Doç. Dr. Ender Mete EKŞİOĞLU (İTÜ)
Diğer Jüri Üyeleri : Yrd. Doç. Dr. İlker BAYRAM (İTÜ)
Yrd. Doç. Dr. Mustafa KAMAŞAK (İTÜ)

HAZİRAN 2011

ÖNSÖZ

Yaptığım çalışmalarda benden desteğini eksik etmeyen ve her konuda bana yardımcı olan hocam Sayın Yrd. Doç. Dr. Ender M. Ekşioğlu'na teşekkürü bir borç bilirim. Ayrıca tüm yüksek lisans çalışmam boyunca benden desteklerini eksik etmeyen ARÇELİK ELEKTRONİK A.Ş. deki tüm yöneticilerime sonsuz teşekkürlerimi sunarım.

Son olarak, başta ailem olmak üzere bana güvenip maddi ve manevi desteğini benden esirgemeyen ve yüksek lisans öğrenimimi bitirmeme yardımcı olan herkese teşekkür ederim.

Mayıs 2011

Emrah YAVUZ

Telekomünikasyon Mühendisi

İÇİNDEKİLER

KISALTMALAR

OMP	**:** Dik Eşleşme Takip Algoritması
BP	**:** Temel Takip Algoritması
FOCUSS	**:** Odaksal Düşük-Tanımlı Sistem Çözümü
MAP	**:** En Büyük Olasılık Metodu
MOD	**:** Optimal Yönler Metodu
K-SVD	**:** K Adet Tekil Değer Ayrıştırması
RLS-DLA	**:** Uyarlamalı Optimal Yönler Metodu
MP	**:** Eşleşme Takip Algoritması
ORMP	**:** Sıralı Yinelemeli Eşleşme Takip Algoritması
PA	**:** Takip Algoritması
SVD	**:** Tekil Vektör Ayrıştırması
SNR	**:** İşaret Gürültü Oranı

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

ÖZET

İŞARETLERİN SEYREK GÖSTERİLİMİ İÇİN SÖZLÜK TASARIMI

Günümüze kadar ses, metin, görüntü ya da video işaretlerinin iletimi için çeşitli teknikler geliştirilmiştir. Tüm bu yöntemlerin üzerinde çalıştığı konulardan biri, haberleşme için kısıtlı bir kaynak olan band genişliğinin verimli bir şekilde kullanılması olmuştur. Mevcut band genişliğinin verimli bir şekilde kullanılması için bir yaklaşım işaretlerin çeşitli sıkıştırma algoritmaları ile boyutlarının küçültülmesidir. Son yıllarda işaretlerin seyrek gösterilimlerinin bulunarak ifade edilmesi bu algoritmalar arasında önemli bir yer tutmaya başlamıştır. Özellikle geçtiğimiz son on yıl içinde, işaretlerin seyrek gösteriliminin bulunması ve kullanılmasına karşı gelişen bir ilgi mevcuttur.

Aşırı-tam bir sözlük yardımıyla bir işaretin seyrek gösteriliminin bulunması kullanılan en yaygın yöntemlerden biridir. Sözlük, atom olarak da adlandırılan ve işaret boyutuna göre daha fazla sayıda olan temel işaretlerin bir kümesidir. Sıkıştırma problemleri dışında işaretlerin seyrek gösterilimi, ters problemlerin düzenlenmesi, gürültü azaltma ve öznitelik çıkarma gibi alanlarda da kullanılmaktadır.

İşaretlerin seyrek gösterilimi için kullanılan sözlük tanılama algoritmalarının şimdiye kadar sunulan türlerinde, sözlüğün bulunması sözlüğün bir grup eğitim vektörünün tümü kullanılarak güncellenmesi şeklinde gerçekleştirilmiştir. Bunun yanında eğitim seti uyarlamalı bir şekilde kullanılarakta sözlük oluşturulabilir.

Bu yüksek lisans tez çalışmasında, günümüzde güncel bir araştırma konusu olan sözlük tanılama yöntemleri ile sinyallerin seyrek gösterilimi konusu üzerinde çalışılmıştır. Bu amaçla, sözlük öğrenme yöntemleri incelenmiş ve çeşitli algoritmalar sinyallerin seyrek gösterilimi için MATLAB ortamında gerçekleştirilmiştir. Farklı algoritmalar için elde edilen sonuçlar arasındaki başarım farklılıkları raporlanmıştır.

SUMMARY

DICTIONARY DESIGN FOR SPARSE SIGNAL REPRESENTATION

Until now, a variety of techniques have been developed for the transmission of audio, text, images or video signals. The efficient use of the bandwidth, which is a limited resource for communication is one of the main issues in any communication context. An approach for the effective employment of existing bandwidth is downsizing the signal dimensionality via different compression algorithms. In recent years, expression of signals with sparse representation gained importance among these compression algorithms. Especially in the past ten years, there is a growing research interest for sparse representation of signals.

Sparse representation of a signal by adopting an over-complete dictionary is one of the common methods used. A dictionary can be defined as a set of basic signals, and the dictionary size or the number of these basic signals is bigger than the dimension of the signal to represent. Other than compression problems, sparse signal representation is also used in areas such as regularization of inverse problems, noise reduction and feature extraction.

Dictionary leaning algorithms for sparse representation generally update the learned dictionary utilizing a whole group of training vectors in each iteration. However, there are recent dictionary leaning algorithms proposals, which update the dictionary estimate in an adaptive fashion rather than using a batch approach.

In this master thesis, a current research topic, dictionary learning algorithms for sparse representation of signals is studied. For this purpose, dictionary learning algorithms are investigated and various algorithms are implemented with MATLAB for the representation of the signals. Differences in performance between the results obtained for different algorithms are reported.

1. GİRİŞ

İşaret işleme alanında yapılan çalışmaların bir bölümü, bir işaretin uygun bir dönüşüm yöntemi kullanılarak gösterilimine odaklanmaktadır. Ayrık ya da sürekli zamanlı işaretlerin "Fourier Dönüşümü" ya da "Dalgacık (Wavelet) Dönüşümü" gibi dönüşüm kuralları ile farklı bölgelere taşınarak ifade edilmesi gösterilim yöntemlerinden birkaçıdır. Son yıllarda yapılan çalışmalar ise işaretlerin bu dönüşüm kuralları dışında bir sözlük yardımıyla seyrek gösterimlerinin bulunarak ifade edilmelerine yönelmiştir. Sıkıştırma, gürültü azaltma, haberleşme ve daha birçok alanda işaretlerin seyrek gösterilimi tercih edilir bir dönüşüm kuralı olmaktadır [1].

İşaretlerin seyrekliği dik bir gösterilimde katsayıların sınırlandırılması yardımıyla sağlanabilir. Bu gerçeklemeye verilebilecek iyi örnekler ayrık kosinüs dönüşümü ("Discrete Cosine Transform") ve "Dalgacık Dönüşümü" şeklindedir [1, 2]. Bunun yanında işaretlerin seyrek gösterilimi için aşırı-tam (over-complete) veya diğer bir deyişle gereksizlik (redundancy) içeren sözlüklerin kullanılması oldukça uygundur. Sözlüğün $\{d^{(k)}\}_{k=1\dots K}$ atomlarının birleşiminden oluştuğu göz önüne alınırsa, aşağıda verilen denklemde bir işareti temsil eden seyrek gösterilimin ifadesi sunulmaktadır.

$$\tilde{x} = \sum_{k=1}^{K} w(k) d^{(k)} \qquad (\mathbf{1.1})$$

(1.1) denkleminde yer alan $\tilde{x}$, seyrek gösterilimi ifade edilmeye çalışılan işareti belirtmektedir. $w(k)$ ise sadece belirli değerleri sıfırdan faklı olan seyrek katsayı vektörünü göstermektedir. $\tilde{x}$ işaretinin seyrek olması bu nedenden ileri gelmektedir. $\tilde{x}$ işareti için seyrek gösterilim probleminde amaç, $d^{(k)}$ atomları bilindiğinde seyrek w katsayı vektörünün bulunmasıdır. Daha zorlu bir problem ise, seyrek gösterilime imkân verecek bir sözlüğün hesaplanmasıdır. Bu problem "sözlük öğrenme" veya "sözlük tasarımı" olarak adlandırılabilir. Sözlük öğrenme probleminin çözümü, seyrek katsayı matrisinin bulunması ve katsayı matrisleri kullanılarak sözlük matrisinin güncellenmesi şeklinde iki aşamalı bir yaklaşımla sağlanabilir. Birinci

aşamada seyrek katsayı matrisi; bir önceki aşamada güncellenen sözlük matrisi ve seyrek gösterilimi bulunmaya çalışılan işaret yardımıyla vektör seçim algoritmalarından biri kullanılarak hesaplanır. Vektör seçim algoritmalarından bir kaçı Dik Eşleşme Takip Algoritması ("Orthogonal Matching Pursuit - OMP") [4,5], Temel Takip Algoritması ("Basis Pursuit - BP") [6] ve Odaksal Az-kararlı Sistem Çözümü ("Focal Under-determined System Solver – FOCUSS") [7] şeklindedir. Bu yöntemlerden her hangi biri kullanılarak belirli sayıda değeri sıfırdan farklı olan katsayı matrisi oluşturulabilir.

İkinci aşamada birden fazla sayıda elemana sahip bir işaret kümesi için uygun olan sözlük matrisi ise birçok yolla tasarlanabilir. Bunlardan bazıları temel fonksiyonlar, örneğin "Gabor" ve "Dalgacık" fonksiyonları kullanılarak sağlanabilir [1]. Bu tip sözlükler literatürde analitik sözlükler olarak adlandırılmaktadır [8]. Bunun yanında belirli bir sınıf işaret için kullanılacak sözlükler yine bu işaretler kullanılarak öğrenme yöntemi ile özel olarak hesaplanabilir. Öğrenme genellikle bir eğitim kümesi temel alınarak sözlüğün eğitim süreci sonunda oluşturulması şeklindedir. Buna karşılık bu tasarım sözlüğün boyutu, yapısı ve gerçekleme şartlarına göre bazı kısıtlara sahiptir. Günümüze kadar birçok sözlük tanılama algoritması sunulmuştur. Bu algoritmalardan bazıları En Büyük Olasılık Metodu ("Maximum Likelihood Methods – MAP") [9, 10, 11], Optimal Yönler Metodu ("Method of Optimal Directions – MOD") [12, 13, 14] ve K adet Tekil Değer Ayrıştırması ("K-Singular Value Decomposition - K-SVD") [2, 3, 11] şeklindedir. Bu yüksek lisans tez çalışmasında ise bu yöntemlere ilave olarak çok daha az iterasyon sonucunda sözlük matrisinin elde edilmesini sağlayan Uyarlamalı Optimal Yönler Metodu ("Recursive Least Squares Method of Optimal Directions - RLS-DLA") [1] tanıtılacak ve gerçeklenecektir.

Yüksek lisans tez çalışmasında, sözlük tanılama yöntemleri kullanılarak işaretlerin seyrek gösterilimi konusu üzerinde çalışılmıştır. Bu amaçla, sözlük tanılama yöntemleri ve vektör seçim algoritmaları incelenmiştir. İşaretlerin seyrek gösterilimi için günümüzde kullanılan ve sözlük tasarımına çok daha hızlı yakınsamak için önerilen yöntemler MATLAB ortamında gerçeklenmiştir. Elde edilen sonuçlar yöntemlerin başarımı ve değişkenlerin etkisi göz önüne alınıp karşılaştırılarak sunulmuştur.

Tez çalışması altı ana bölümden oluşmaktadır. Çalışmanın bu giriş bölümünde işaretlerin seyrek gösterimi genel bilgiler çerçevesinde tanıtılmıştır. Bölüm içinde seyrek gösterim için kullanılan sözlük tanılama kavramı ve yöntemleri hakkında kısa bilgiler verilmiştir. İkinci bölümde, seyrek işaret işleme, işaretlerin seyrek gösterimleri için kullanılan vektör seçim algoritmaları ve sözlük tanılama kavramları hakkında daha detaylı bilgiler sunulmuştur.

Çalışmanın üçüncü ve dördüncü bölümünde seyrek gösterilime imkan verecek sözlük tanılama algoritmaları açıklanmıştır. Üçüncü bölümde bu algoritmaların geleneksel olarak tanımlayabileceğimiz ve veri üzerinde bloklar şeklinde işlem gerçekleştirilen çeşitleri açıklanmıştır. Dördüncü bölümde ise sonuca yani sözlük matrisine çok daha az iterasyonda yakınsayan uyarlamalı yöntemler verilmiştir.

Beşinci bölümde bahsedilen tüm sözlük tanılama algoritmaları MATLAB ortamında gerçekleştirilmiş ve sonuçlar karşılaştırılarak raporlanmıştır. Bu sonuçlar problemde değişken olarak yer bulan sözlük matrisinin sütun veya atom sayısı, veri kümesinin boyutu, işaretin uzunluğu, seyreklik değeri gibi faktörler değiştirilerek incelenmiştir. Çalışmanın son bölümde çıkan sonuçlar tartışılarak tez çalışması tamamlanmıştır.

2. SEYREK İŞARET İŞLEME

Tez çalışmasının bu bölümünde işaretlerin seyrek gösterilimi için kullanılan vektör seçim algoritmaları ile birlikte seyrek gösterilim kavramı hakkında detaylı bilgiler verilecektir.

2.1 Seyrek Gösterilim Kavramı

Son zamanlarda işaretlerin seyrek gösterimine karşı yapılan araştırmalarda artan bir ilgi olduğu görülmektedir [1, 2, 11, 15, 16]. $x \in \mathbb{R}^M$, işareti, aşırı-tam bir sözlük matrisinin atomlarının seyrek, lineer (doğrusal) birleşimi şeklinde gösterilebilir. Kısa bir ifade ile seyrek gösterilim, bir x işaretinin, D sözlüğü ve w seyrek gösterim katsayıları kullanılarak lineer bir birleşim ile ifade edilmesidir. Bu ifadede yer alan sözlük matrisi $D \in \mathbb{R}^{M \times K}$ şeklinde bir matrisle gösterilmekte ve K adet atomdan oluşmaktadır. Bu durumda x işareti; D sözlüğü yardımıyla, $||x - Dw||_p \leq \varepsilon$ şartını sağlanmak üzere $x = Dw$ kesin şekilde ya da $x \approx Dw$ yaklaşık şekilde ifade edilebilir. $w \in \mathbb{R}^K$, ağırlık vektörü olarak isimlendirilmekte ve x işaretine ait katsayıları göstermektedir. Ağırlık vektörünün sadece belirli sayıda elemanının sıfırdan farklı değer alması x işaretinin seyrek gösterilimini sağlamaktadır. Varsayılan seyrek gösterilim metotlarında, seyrekliği ölçmek için çeşitli sayılar kullanılmaktadır [3]. Sayılar sıfır alınarak bir işaretin seyrek gösterilimin ifadesi denklem (2.1)'de verilmiştir.

$$(P_0) \qquad \min_w \left|\left|w\right|\right|_0 \qquad x = Dw \qquad (\mathbf{2.1})$$

w katsayı vektörüne ya da en genel şekli ile bir işarete ait l^p sayısı; $\|w\|_p = (\sum_{k=1}^{K} |w_k|^p)^{1/p}$ şeklinde ifade edilebilir. Bu durumda $\left|\left|\cdot\right|\right|_0$ argüman vektörün sıfır sayısını (l^0 - count) ifade etmektedir. Sıfır sayısı vektörün sıfırdan farklı elemanlarının sayısını vermektedir. O halde, denklem (2.1) ile bir *x* vektörünü, *D* matrisinin atomları cinsinden ifade eden en seyrek w ağırlık vektörünün hesaplanması tanımlanmaktadır. Uygulamada denklem (2.1) ile ifade edilen seyrek gösterilimi sağlamak üzere çeşitli takip algoritmaları mevcuttur [4, 5, 11]. Bu

algoritmalar işaretlerin seyrekliğini tam ya da yaklaşık olarak sağlamaktadır. Bölümün alt başlıklarında bu algoritmaların bir kaçı açıklanacak ve başarımları değerlendirilecektir. Seyrek gösterilim veya diğer bir deyişle seyrek kodlama, işaretlerin seyrek gösterimi için kullanılacak sözlüklerin tasarımında gerekli bir adımdır. Bu tezde sözlük tasarımı inceleneceğinden, seyrek gösterilim problemine de değinilmektedir.

Seyrek gösterilimde kullanılan algoritmalardan bazıları Eşleşme Takip Algoritması ("Matching Pursuit - MP") [4], Dik Eşleşme Takip Algoritması ("Orthogonal Matching Pursuit - OMP") [4, 5] ve Sıralı Yinelemeli Eşleşme Takip Algoritması ("Order Recursive Matching Pursuit - ORMP") [4, 5, 11] şeklindedir. Gerçekleme yöntemleri basit olan bu algoritmaların seyrek gösterilimi oldukça başarılı bir şekilde sağladığı söylenebilir. Tüm bu algoritmalar işaretler ile sözlük elemanları arasındaki içler çarpımı ve en az kareler yöntemi gibi işlemleri içermektedir.

Diğer bir iyi bilinen seyrek gösterilim yaklaşımı, Temel Takip ("Basis Pursuit - BP") [6] algoritmasıdır. Bu yöntemde kestirim hatasını ölçmek için l^0 sayısı yerine l^1 kullanılması önerilmektedir. Benzer şekilde Odaksal Az-kararlı Sistem Çözümü ("Focal Under-determined System Solver – FOCUSS") [7], algoritmasında l^0 sayısı yerine, $p \leq 1$ olmak üzere l^p sayısının kullanması önerilmektedir. Alt bölümlerde bu algoritmalar açıklanmıştır.

2.2 Dik Eşleşme Takip algoritması ("Orthogonal Matching Pursuit - OMP")

Dik eşleşme takip algoritması ("Orthogonal Matching Pursuit - OMP"), adım adım güncelleme ile gerçekleştirilen bir seyrek gösterilim algoritması türüdür [4, 5]. Bu algoritmanın her bir adımında sözlük elemanları, fark işarete (kestirim hatasına) karşı en büyük gösterimi sağlayacak şekilde seçmektedir. Bu durum sözlük elemanlarının ya da sözlük matrisinin normalize edildiği varsayımı altında gerçekleştirilmektedir. Her bir sözlük elemanı seçiminden sonra, katsayılar (ağırlık) vektörü en küçük kareler yöntemi kullanılarak hesaplanır. $x \in \mathbb{R}^M$ veri kümesini, D K adet normalize edilmiş sütuna sahip sözlük matrisini göstermek üzere algoritmanın adımları aşağıdaki şekilde tanımlanabilir [3, 4, 5, 11]. Başlangıç koşullarında $r_0 = x$ ve $k = 1$ olmak üzere kabul edilirse:

1. Yeni seçilecek sözlük atomunun indeks değerinin belirlenmesi, $i_k = argmax_v |\langle r_{k-1}, d_v \rangle|$

2. Mevcut kestirimin güncellenmesi, $x_k = argmin_{x_k} ||x - x_k||_2^2$, $x_k \in span\{d_{i1}, d_{i2}, \dots, d_{ik}\}$

3. Fark değerin (hatanın) güncellenmesi, $r_k = x - x_k$

Algoritma önceden belirlenen bir iterasyon değeri sonucu durdurulabilir. Diğer bir yöntem olarak iterasyon sonlarında fark değerinin (hatanın) alacağı belirli bir değere kadar devam ettirilebilir [11]. Her iki yöntemde uygulamada kullanılan yöntemlerdir. Ancak bu çalışmada iterasyonun belirli bir adımda durdurulması kullanılmıştır. Dik eşleşme takip algoritması (OMP) gerçeklemesi oldukça basit bir algoritmadır. BP benzeri yöntemlerin aksine, bu yöntemde belirli bir seyreklik değeri için istenilen katsayı vektörlerinin hesaplanması sözlük eğitim kümesi (sözlük matrisi) yardımıyla kolayca programlanabilir [4, 5]. Bu algoritmanın birçok çeşidi bulunmaktadır. Bunlardan bazıları; (i) en az kareler yönteminin atlanması ve içsel çarpımların katsayı olarak kullanılması, (ii) sözlüğün her bir aday atomu için en az kareler yönteminin uygulanması ve sadece seçim aşamasında içsel çarpımların değerlendirilmesi, (iii) her bir yeni atom seçimi öncesi, seçilen atomlar yardımıyla seçilmeyenleri kapsayacak şekilde tasarlanması, (iv) algoritmayı hızlandırmak üzere, en büyük içsel çarpımı bulmak yerine en yakın en büyük olanın seçilmesi şeklindedir [3].

2.3 Temel Takip Algoritması ("Basis Pursuit - BP")

Temel takip algoritması ("Basis Pursuit - BP"), dik eşleşme takip algoritmasından farklı olarak, kestirimde l^0 sayılarının l^1 sayıları ile yer değiştirmesine dayanmaktadır [3, 6]. Bu durumda seyrek kodlama çözümleri denklem (2.2) ve denklem (2.3)'de verildiği şekli ile yazılabilir. Denklem (2.2) kesin çözümü ifade ederken denklem (2.3) seyrek gösterilimde katsayıların hesaplanması için yaklaşık çözüm sonucunu vermektedir. Denklem (2.3) ve denklem (2.4)'de yer alan x seyrek gösterilimi bulunmaya çalışılan işareti, D ise sözlük matrisini göstermektedir. w, seyrek ağırlık vektörünü belirtmektedir.

$$(P_1) \qquad \min_w \left|\left|w\right|\right|_1 \qquad x = Dw \tag{2.2}$$

$$\left(P_{1,\varepsilon}\right) \qquad \min_w ||w||_1 \qquad \left|\left|x - Dw\right|\right| \leq \varepsilon \tag{2.3}$$

Yukarıdaki denklemlerde yer alan $\left|\left|\cdot\right|\right|_1$ ifadesi bir işaretin ya da vektörün birinci dereceden sayısını ifade etmektedir. Açık bir ifade ile $\left|\left|w\right|\right|_1$ sayısı; $\left|\left|w\right|\right|_1 = \sum_{k=1}^{K}|w_k|$ şeklinde yazılabilir. İfadeden de görüleceği gibi $\left|\left|\cdot\right|\right|_1$ sayısı bir vektörün değerleri toplamının mutlak değerini vermektedir.

Seyreklik problemi için kesin ya da yaklaşık gösterimlerin (2.2, 2.3) her ikisinde de lineer programlama verimli çözümler sağlanmaktadır. Diğer yandan l^1 sayısı l^0 sayısının konveks bir yaklaşıklığı olarak görülebilir [11]. Bu durumda, denklem (2.2) ifadesi birleştirilmiş formda denklem (2.4)'de verildiği şekli ile yazılabilir. Denklemde yer alan τ terimi regülasyon faktörü olarak adlandırılmakta ve seyreklik değerinin belirlenmesini etkilemektedir [6]. Bu değer 1 den büyük değerlere sahip olmakla birlikte seyreklik değerini direkt etkilemesi terimin değerini seçilmesini zorlaştırmaktadır.

$$\min_{w} \frac{1}{2}||Dw - x||_2^2 + \tau||w||_1 \qquad (\mathbf{2.4})$$

Temel Takip algoritması; $x \in \mathbb{R}^M$ bir işareti, $D \in \mathbb{R}^{M\times K}$ sözlük matrisini göstermek üzere aşamalı şekilde de ifade edilebilir [3]. Bu gösterimde regülasyon terimi $\tau \geq 1$ ve başlangıç katsayı vektörü w_0 olarak alınmıştır. Elde edilmek istenen çıktı katsayı (ağırlık) matrisinin bulunmasıdır.

1. Başlangıç değeri olarak k=1 ile iterasyona başla.
2. $a_k \geq 0$ seçilmesi ve denklem (2.5) yardımıyla w_k^+ hesaplanması.
3. $\gamma_k \in (0,1]$ şeklinde seçilmesi ve denklem (2.6) yardımıyla w_{k+1} ifadesinin sağlanması.
4. Algoritmanın durma ölçütünün kontrolü, eğer şart sağlanmaz ise k teriminin 1 arttırılarak iterasyona devam edilmesi

$$w_k^+ = argmin_z (z - w_k)^* D^* (Dw_k - x) + \frac{1}{2} a_k ||z - w_k||_2^2 + \tau\left|\left|z\right|\right|_1 \qquad (\mathbf{2.5})$$

$$w_{k+1} = w_k + \gamma_k (w_k^+ - w_k) \qquad (\mathbf{2.6})$$

Denklemlerde z terimi $-z \leq w \leq z$ şeklinde ifade edilen bir terimdir.

2.4 Odaksal Az-kararlı Sistem Çözümü ("Focal Under-determined System Solver – FOCUSS")

Odaksal Az-kararlı Sistem Çözümü algoritması, ("Focal Under-determined System Solver – FOCUSS") denklem (2.2) ve denklem (2.3)'deki çözümleri bulmak için kullanılan ve $p \leq 1$ olmak üzere l^0 sayısı l^p sayısı ile değiştiren bir algoritmadır [3, 7]. Bu durumda kesin çözüm için yazılan denklem (2.2) aşağıda ifade edilen denklem (2.7) durumuna gelecektir. Denklemde yer alan x seyrek gösterilimi bulunmaya çalışılan işareti, D sözlük matrisini belirtmektedir. w ise seyrek ağırlık vektörünü göstermektedir.

$$(P_p) \qquad \min_w ||w||_p^p \qquad x = Dw \tag{2.7}$$

Katsayı vektörünün bulunması için kullanılan FOCUSS algoritmasının adımları bölümün geri kalan kısmında açıklanmıştır. Lagranj çoğullama vektörü $\lambda \in \mathbb{R}^M$ kullanılmak üzere, Lagranj fonksiyonu denklem (2.8) verilen şekli ile yazılabilir.

$$L(w, \lambda) = ||w||_p^p + \lambda^T (x - Dw) \tag{2.8}$$

Bir katsayı vektörü w için gerekli koşulları sağlamak üzere, λ çoğullama vektörü çözümü denklem (2.9)'da verildiği gibi olmalıdır.

$$\nabla_w L(w, \lambda) = p \prod(w) w - D^T \lambda = 0 \quad ve \quad \nabla_\lambda L(w, \lambda) = Dw - x = 0 \tag{2.9}$$

Denklemde yer alan $\prod(w)$ çarpımı bir diyagonal matristir ve elemanları $|w_i|^{p-2}$ olmak üzere (i, i) boyutundadır. Bir çok cebirsel işlem sonucunda denklem (2.9), aşağıda ifade edilen denklem (2.10)'da yer alan çözümü verecektir.

$$w = \prod(w)^{-1} D^T (D \prod(w)^{-1} D^T)^{-1} x \tag{2.10}$$

İfade edilen denklemden görüleceği üzere w katsayılarının bulunması için kapalı çözüm mümkün gözükmemektedir. Bu nedenle yinelemeli bir çözüme ihtiyaç duyulmaktadır. Bu güncellemeli çözüm denklem (2.11)'de verilen şekli ile yazılabilir [5]. Denklemin sağ tarafında katsayı vektörünün bir önceki bilinen sonucu yer almaktadır. Denklem (2.11)'de verilen yinelemeli süreç ile katsayı vektörünün, w, (ağırlık vektörünün) elemanlarına ulaşmak mümkündür.

$$w_k = \prod(w_{k-1})^{-1} D^T (D \prod(w_{k-1})^{-1} D^T)^{-1} x \tag{2.11}$$

Temel takip algoritması (“BP”) [4,5] ve Odaksal Az-kararlı Sistem Çözümü (“FOCUSS”) [6] sözlük tanılama yöntemlerinde biri olan En Büyük Olasılık Metodu (“Maximum Likelihood Methods – MAP”) [9, 10] yönteminde sıklıkla kullanılmaktadır. Ancak bu tez çalışmasında gerçeklenen sözlük öğrenme algoritmalarının tümünde, seyrek gösterilim adımında Dik Eşleşme Takip algoritması (“Orthogonal Matching Pursuit - OMP”) vektör seçim algoritması olarak tercih edilmiştir.

3. SÖZLÜK TASARIMINDA KULLANILAN YÖNTEMLER

Tez çalışmasının bu bölümünde sözlük tanılama kavramı hakkında bilgi verilerek işaretlerin seyrek gösteriliminde kullanılan blok güncellemeli geleneksel sözlük tanılama yöntemleri tanıtılacaktır.

3.1 Sözlük Tanılama

İşaretlerin seyrek gösterilimi için kullanılan bir sözlük, atom adı verilen sütun vektörlerinin birleşmesinden meydana gelmiştir [17, 18]. Sözlük içindeki atomların bir kısmının lineer birleşimi, orijinal işareti tam ya da yaklaşık bir gösterilimini ifade etmede kullanılabilir. Bu ifade daha önce tanımlanan ve denklem (2.1) ile ifade edilen seyrek gösterime eş değer bir ifadedir. Genellikle bir işareti ifade etmek için bir sözlük ve bir çerçeve aynı kavram olarak görülebilir, ancak iki kavram arasında küçük bir farklılık mevcuttur. Bir çerçeve işaret uzayını kapsamaktadır, ancak sözlük için aynı durumu söylemek mümkün değildir [1, 19].

Seyrek gösterilimi bulunacak orijinal işaret reel elemanlı ve uzunluğu M olan bir x vektörü ile gösterilebilir. Sözlük ise bir vektörün uzunluğu M olmak üzere K adet sütun vektöründen diğer bir ifadeyle atomdan $\{d^{(k)}\}_{k=1}^{K}$ oluşmaktadır. Her bir atom M adet elemana sahiptir. Bu durumda D sözlüğü $D \in \mathbb{R}^{M \times K}$ şeklinde bir matris biçiminde gösterilebilir ve bu sözlüğün her bir sütun vektörü; bir atom şeklinde ifade edilir. Tüm bu bilgiler ile birlikte x işaretinin D sözlüğü ile kesin ya da yaklaşık seyrek gösterilimi ve bu gösterilimden ortaya çıkan kestirim hatası, r, (3.1) ve (3.2) numaralı denklemlerinde verildiği şekilde olacaktır.

$$\tilde{x} = \sum_{k=1}^{K} w(k) d^{(k)} = Dw \qquad (\mathbf{3.1})$$

$$r = x - \tilde{x} = x - Dw \qquad (\mathbf{3.2})$$

Denklemlerde yer alan $w(k)$, k atomu için yani k. sözlük sütunu için kullanılan katsayı değerini ifade etmektedir. Denklemde de ifade edildiği üzere $\{w(k)\}_{k=1}^{K}$

katsayılarının birleşmesi sonucu K uzunluklu w katsayı vektörü elde edilmektedir. Bu vektörün en genel hali denklem (2.1)'de verilen seyreklik şartını sağladığı ve sadece sonlu sayıda elemanının sıfırdan farklı değer aldığı söylenebilir.

Seyrek gösterilimde sadece az sayıdaki değerin (s tane), sıfırdan farklı bir değer almasına izin verilmektedir. Örneğin s değerinin 4 olduğu bir durumda w vektörünün 4 elemanı dışındaki tüm değerler sıfıra eşittir. Sıfır dışındaki değerler vektörün herhangi bir indisli elemanına ait olabilir. Buradan s/M terimini seyreklik faktörü şeklinde tanımlamak mümkündür [1].

Bu bölüme kadar açıklanan denklem (3.1) ve denklem (3.2) sadece M veri uzunluğuna sahip bir vektörün seyrek ifadesi verilmiştir. Bunun yanında M uzunluklu N adet x vektörünün birleşmesi ile oluşan $X = \{x_i\}_{i=1}^{N}$ matrisinin, D sözlüğü yardımıyla seyrek gösterilimini gerçekleştirmekte mümkündür. Bu durumda kullanılması gereken W katsayı matrisi de $W \in \mathbb{R}^{K \times N}$ şeklinde olacaktır. En genel şekli ile çözülmesi gereken seyrek gösterilim ifadesi $X = DW$ şeklinde ifade edilebilir.

Sözlük öğrenme algoritmalarında gerçeklenecek temel işlev sözlüğün bir eğitim kümesinden yola çıkılarak belirlenmiş bir iterasyon ya da hata değerine kadar eğitilmesi ya da diğer bir ifadeyle güncellenmesidir. Seyrek gösterilim probleminde eğitim kümesi olarak seyrek gösterilimi bulunacak işaret matrisi X kullanılmaktadır. Güncelleme sırasında gerçeklenen genel yöntem eğitim kümesinin seyrek gösterilimi yardımıyla toplam hatanın, karesel hatanın, mümkün olduğunca küçük değerlere çekilmesidir. X matrisinin sütunları eğitim vektörlerini ve W matrisinin sütunları seyrek katsayı vektörlerinin göstermek üzere karesel hatanın küçültülmesi $f(.)$ fonksiyonu yardımıyla denklem (3.3)'de verilen şekilde olacaktır. Bunun yanında seyreklik şartını göz önünde bulundurmak üzere denklem (3.4) ve denklem (3.5)'de hesaplanmalıdır. Denklemlerde yer alan D, sözlük matrisini belirtmektedir.

$$argmin_{D,W} f(D,W) = argmin_{D,W} \sum_i \left|\left|r_i\right|\right|_2^2 = argmin_{D,W} \sum_i ||X - DW||_F^2 \qquad (\mathbf{3.3})$$

$$\min_{D,W} \{||X - DW||_F^2\}, \qquad \forall i, ||w_i||_0 \leq T_0 \qquad (\mathbf{3.4})$$

$$\min_{D,W} \sum_{i=1} ||w_i||_0, \qquad ||X - DW||_F^2 \leq \varepsilon \qquad (\mathbf{3.5})$$

Denklem (3.3)'de yer alan $X \in \mathbb{R}^{M \times N}$ uzunluklu bir matris olup daha açık bir ifade ile $X = \{x_i\}_{i=1}^{N}$ şeklinde yazılabilir. Benzer şekilde W katsayı matrisi de $W \in \mathbb{R}^{K \times N}$ uzunluğuna sahiptir ve $W = \{w_k\}_{i=1}^{N}$ şeklinde ifade edilebilir. Aynı zamanda denklem (3.3), denklem (3.4) ve denklem (3.5)'de ifade edilen $||A||_F^2$ notasyonu Frobenius norm olarak adlandırılmakta ve $||A||_F = \sqrt{\Sigma_{ij} A_{ij}^2}$ şeklinde ifade edilmektedir.

Denklem (3.3)'den işaret için kestirilen matris $\tilde{X} = DW$ şeklinde ve kestirim hatası $R = X - \tilde{X} = X - DW$ şeklinde ifade edilebilir. R hata terimi aynı zamanda $R = \{r_i\}_{i=1}^{N}$ birleşimden oluştuğu söylenebilir.

(3.3), (3.4) ve (3.5)'in ifade ettiği ve sözlük öğrenme için tanımlanan optimizasyon problemleri, iki tane matris değişken (D ve W) üzerinden optimizasyon gerektirmektedir. Bu işlemi gerçekleştirmenin zorluğundan dolayı problem güncellemeli iki bölüme ayrılarak ifade edilebilir. Bu iki parça,

1) D matrisinin sabit tutularak W katsayılarının bulunması
2) W katsayılarını sabit tutularak D matrisinin bulunması

şeklinde iki bölüme ayrılabilir [1]. Bu iki parçanın ilk bölümünde D sözlük matrisi sabit tutulmakta ve katsayılar ya da diğer bir ifadeyle ağılıklar vektörü seyreklik faktörü kullanılarak bazı kurallar çerçevesinde (vektör seçim algoritmaları ile) hesaplanmaktadır. Vektörlerin seçilmesi problemini denklem (3.6) yardımıyla ifade edebiliriz. w_{opt} , değeri hesaplanan optimum seyrek katsayı vektörünü göstermektedir.

$$w_{opt} = argmin_w ||x - Dw||_2 \qquad ||w||_0 \leq s \qquad (\mathbf{3.6})$$

Yukarıdaki denklemde mevcut olan $||.||_0$ sembolü, yani l^0 sayısı, bir katsayı vektörü içinde kaç adet elemanın sıfırdan farklı olabileceğini yani seyrekliği göstermektedir. Bir katsayı vektörü $W = \{w_k\}_{i=1}^{N}$ matrisinin bir sütunu olarak görülmektedir. Pratik uygulamalarda w_k ağırlık vektörlerinin, katsayılarının bulunması için iyi sonuç veren birkaç vektör seçim algoritması mevcuttur, bu algoritmlardan bir kaçı bölüm 2'de sunulmuştur. Bu algoritmalara Dik Eşleşme Takip Algoritması ("Orthogonal Matching Pursuit - OMP") [4, 5, 11], Sıralı Yinelemeli Eşleşme Takip Algoritması ("Order Recursive Matching Pursuit - ORMP") [11], Temel Takip Algoritması

("Basis Pursuit - BP") [6] ve Odaksal Az-kararlı Sistem Çözümü ("Focal Under-determined System Solver – FOCUSS") [7] örnek olarak verilebilir. Sadece belirli değerleri sıfırdan farklı olan katsayı vektörlerinin bulunması için bahsedilen vektör seçim algoritmalarından biri kullanılabilir. Bu tez kapsamında OMP algoritması kullanılmıştır.

Güncellemenin ikinci ve daha basit olan kısmında, W katsayı matrisi sabit tutulurken, D sözlük matrisinin güncellenmektedir. Bu işlem Optimal Yönler Metodu ("Method of Optimal Directions – MOD") [12, 13, 14] ve K adet – Tekil Değer Ayrıştırma (K-Singular Value Decomposition - K-SVD) [2, 3, 11] gibi algoritmalar kullanılarak farklı şekillerde gerçekleştirilebilir. Bu tez kapsamında ayrıca Uyarlamalı Optimal Yönler Metodu ("Recursive Least Squares Method of Optimal Directions - RLS-DLA") [1] yöntemi tanıtılacaktır.

MOD algoritması en az kareler çözümünü kullanmakta ve sözlük sözlük matrisinin elde edildiği ikinci aşamada $f(D) = ||X - DW||_F^2$ fonksiyonunu azaltacak şekilde hesaplama işlemi yapılmaktadır. Hesaplamanın ikinci bölümünde KSVD algoritması ise faklı bir yaklaşım ile D sözlük matrisine ulaşmaktadır [3]. Bu yöntemde sözlük atomlarının sıfırdan farklı girişleri sabit tutularak sözlük matrisi güncellenmektedir. MOD ve K-SVD yöntemleri ikinci adımda sözlük matrisinin bulunması aşamasında uyguladıkları yöntemler açısından farklılık göstermektedir.

Denklem (3.3) seyrek gösterilimde karesel hatanın kontrolünü sağlarken, denklem (3.6) seyreklik koşulu ile ilgili kısıtı göstermektedir. Bu denklemler ile verilen kısıtlamalar, birçok sözlük öğrenme yaklaşımlarından sadece birkaçının ortaya çıkmasını sağlamaktadır. Literatürde mevcut diğer denklem, fonksiyon ve yöntemler başka sözlük öğrenme algoritmalarının ortaya çıkmasını sağlamaktadır [1]. Bunun yanında 1. dereceden sayıların kullanılması vektör seçiminde seyreklik ölçümünün daha verimli yapılmasının sağlanması açısından önemlidir.

3.2 En Büyük Olasılık Metodu ("Maximum Likelihood Methods –MAP")

İşaretlerin seyrek gösterimlerinde kullanılan D sözlüğünün oluşturulmasında gerçekleştirilebilecek yöntemlerden biri olan En Büyük Olasılık Metodu ("Maximum Likelihood Methods –MAP") olasılıksal sonuçları kullanarak D matrisini meydana getirmektedir [9, 10, 11]. Denklem (3.7) seyrek gösterilimin genel ifadesini

vermektedir. Denklemde yer alan x seyrek gösterilimi gerçekleştirilen işareti, v vektörü ise σ^2 varyanslı beyaz, normal dağılımlı gürültüyü göstermektedir. w ise seyrek katsayı vektörlerini belirtmektedir.

$$x = Dw + v \tag{3.7}$$

Örnekler kümesinin birleştirilmesi ile $X = \{x_i\}_{i=1}^{N}$ matrisi oluşmakta ve olasılık fonksiyonunu $P(X|D)$, en büyük yapacak sözlük aranmaktadır. Bu işlem için iki adet varsayım kabul edilmek durumundadır. Birincisi denklem (3.8)'de kolaylıkla ifade edildiği üzere ölçümler bağımsız olarak gösterilebilmelidir.

$$P(X|D) = \prod_{i=1}^{N} P(x_i|D) \tag{3.8}$$

İkinci varsayım ise kritiktir ve "saklı değişken" x 'i göstermektedir. Olasılık fonksiyonunun maddeleri, parçaları denklem (3.9) yardımıyla hesaplanabilir [10].

$$P(x_i|D) = \int P(x_i, w|D)dw = \int P(x_i|w, D)P(w)dw \tag{3.9}$$

Denklem (3.7) ile verilen ilk varsayıma dönersek aşağıdaki denklem (3.10) olasılık fonksiyonunu ifade etmek üzere yazılabilir.

$$P(x_i|w, D) = sbt\,.\exp\left\{\frac{1}{2\sigma^2}\left||Dw - x_i|\right|^2\right\} \tag{3.10}$$

w, vektörünün öncelikli dağılımlarından bir kaçı sıfır ortalamalı Cauchy ve Laplace dağılımları olarak verilebilir. Bu dağılımın Laplace dağılımı olduğu varsayımı altında olasılık fonksiyonu denklem (3.11)'de ifade edildiği şekle dönüşecektir.

$$\begin{aligned} P(x_i|D) &= \int P(x_i|w, D)P(w)dw \\ &= sbt\,.\int \exp\left\{\frac{1}{2\sigma^2}\left||Dw - x_i|\right|^2\right\}.\exp\{\lambda||w||_1\}\,dw \end{aligned} \tag{3.11}$$

Katsayılar vektörü üzerinde yapılan bu işlemin gerçekleştirmesi oldukça zordur ve, [10]'da bu işlemin katsayı vektörleri ile $P(x_i, w|D)$ olasılık fonksiyonunun uç değerini yer değiştirerek gerçeklenmiştir. Bu durumda problem denklem (3.12)'da ifade edildiği şekle dönüşmüştür.

$$D = argmax_D \prod_{i=1}^{N} \max_{w_i} \{P(x_i, w_i|D)\}$$

$$= argmin_D \sum_{i=1}^{N} \min_{w_i} \{\left||Dw_i - x_i|\right|^2 + \lambda||w_i||_1\} \quad (\mathbf{3.12})$$

Denklem (3.12)'nün çözümü için güncellemeli bir metot önerilebilir. Bu durumda her bir iterasyonda iki işlem gerçekleştirilecektir. Birinci işlemde basit bir vektör seçim yöntemi kullanılarak w_i katsayıları hesaplanacak, ikinci işlemde ise sözlük matrisi denklem (3.13)'de verildiği şekli ile güncellenecektir.

$$D^{(n+1)} = D^{(n)} - \eta \sum_{i=1}^{N} (D^{(n)} w_i - x_i) w_i^T \quad (\mathbf{3.13})$$

Bunun yanında Lewicki ve Sejnowski tarafından denklem (3.11)'in gerçekleştirilmesi için farklı yaklaşımlar da önerilmiştir [9].

3.3 Optimal Yönler Metodu ("Method of Optimal Directions – MOD")

Sözlük tanılama algoritmalarından bir diğeri ("Method of Optimal Directions – MOD") olarak adlandırılan ve Engan ve arkadaşları [12, 13, 14, 19] tarafından gerçekleştirilen yöntemdir. Bu yöntem bir sonraki bölümde açıklanacak olan K-SVD algoritması ile özellikle katsayı vektörlerinin hesaplanması aşamasında yakın benzerlik göstermektedir. Ancak iki yöntemin sözlük güncelleme kısımları birbirinden farklıdır. Bunun yanında yüksek lisans tezinin konusu olan uyarlamalı güncellemeli süreçlere geçiş MOD algoritması temel alınarak gerçekleştirilmiştir.

MOD algoritmasının en önemli özelliklerinden biri sözlük güncelleme kısmının yani D matrisinin güncellenmesinin, oldukça basit ve verimli bir yöntem ile gerçekleştirilmesidir. Her bir örnek için seyrek gösterilimin bilindiği varsayımı altında, her bir iterasyondaki kestirim hatası $r_i = x_i - Dw_i$ şeklinde ifade edilebilir. Bu ifade de yer alan i kaçıncı iterasyonun yapıldığını göstermektedir. x, $M \times 1$ uzunluklu veri vektörünü ve w, $K \times 1$ uzunluklu katsayı vektörünü göstermektedir. i. iterasyon için hesaplanan r_i teriminden yola çıkarak, genel gösterimde ortalama karesel hata denklem (3.14)'de verildiği şekilde ifade edilebilir [12].

$$||R||_F^2 = |||[r_1, r_2, \ \dots \ , r_N]|||_F^2 = ||X - DW||_F^2 \quad (\mathbf{3.14})$$

Denklem (3.14)'de tüm x_i veri işaretleri $X = \{x_i\}_{i=1}^{N}$ matrisinin sütunlarını, benzer şekilde w_i katsayı vektörlerinin birleşimi $W = \{w_k\}_{i=1}^{N}$ ağırlık matrisini (seyrek katsayı matrisini) oluşturmaktadır. Bunun yanında matris boyutlarının $X \in \mathbb{R}^{M \times N}$ ve $W \in \mathbb{R}^{K \times N}$ şeklinde olacağı görülmektedir. Aynı zamanda denklem (3.14)'de ifade edilen $||A||_F^2$ notasyonu Frobenius norm olarak adlandırılmakta ve $||A||_F = \sqrt{\Sigma_{ij} A_{ij}^2}$ şeklinde ifade edilmektedir.

W, ağırlık matrisinin sabit olduğu durumda, D sözlük matrisinin güncellenmesi denklem (3.14)'de verilen hata ifadesi minimize edilerek gerçekleştirilir. Hatanın küçültülmesi için denklem (3.14)'de verilen hata denkleminin D matrisine göre türevi alınarak $(X - DW)W^T = 0$ bağlantısı sağlanır ve sözlük güncellemesi için gerekli ifade denklem (3.15)'da verildiği şekilde bulunur. Denklemde yer alan i değişkeni iterasyon indisini göstermektedir.

$$D^{(i+1)} = XW^{(i)^T}.(W^{(i)}W^{(i)^T})^{-1} \qquad (\mathbf{3.15})$$

Denklem (3.15)'de yer alan $W^{(i)}W^{(i)^T}$ ifadesinin her zaman bir tersi bulunacağı açıktır. $W^{(i)^\dagger} = W^{(i)^T}.(W^{(i)}W^{(i)^T})^{-1}$, $W^{(i)}$ matrisinin sözde tersi (pseudoinverse) olarak adlandırılır. (3.15) denklemi $D^{(i+1)} = XW^{(i)^\dagger}$ olarak da yazılabilir. Başlangıç koşullarında sözlük matrisi olarak X veri matrisinin ilk K sütunu seçilebilir. Denklem (3.15) ile verilen güncelleme denklemi sabit bir katsayı matrisi için hesaplanabilecek en iyi ve en hızlı sözlük matrisinin bulunmasını sağlamaktadır. Önceki bölümde açıklanan MAP yöntemi ile sözlüğün bulunması çok daha yavaş olmaktadır [3, 11].

D_0 başlangıç sözlüğünü göstermek üzere Optimal Yönler Metodu (MOD) aşamalı şekilde aşağıdaki şekilde de ifade edilebilir [1]. D_0 başlangıç sözlüğü olarak eğitim kümesinin, yani veri kümesinin, ilk K vektörü alınabilir. j iterasyon indisini göstermektedir. i ise veri kümesindeki sütunların indisini ifade etmekte ve $i = 1 \ldots . N$ arasında değer almaktadır.

1. j. iterasyonda, x_i, i. veri vektörünün alınması. X veri matrisi, $X = \{x_i\}_{i=1}^{N}$ şeklinde N adet sütundan oluşmaktadır ve her bir sütun x_i veri vektörü olarak görülebilir.
2. D_{j-1} sözlük matrisini, , x_i , i . veri vektörünü ve bir vektör seçim algoritmasını (bu çalışma için OMP kullanılmıştır) kullanarak w_i katsayı

vektörlerinin bulunması. Elde edilen w_i vektörü W katsayı matrisinin i. sütununu göstermektedir. Bu adımdaki işlem j. iterasyonda, tüm x_i değerleri alınarak w_i 'lerin hesaplanmasını içerir. w_i vektörlerinin birleşmesi ile $W = \{w_k\}_{i=1}^{N}$ ağırlık matrisi oluşacaktır.

3. j. iterasyonda denklem (3.15) kullanılarak D_j sözlük matrisi X veri ve W ağırlık matrisleri yardımıyla hesaplanır.

$$D^{(j+1)} = XW^{(j)^T}.(W^{(j)}W^{(j)^T})^{-1}$$

4. Hesaplanan sözlüğün normalize edilmesi.

5. j iterasyon değerinin arttırılarak 1. adıma geri dönülmesi.

3.4 K adet – Tekil Değer Ayrıştırma (K-Singular Value Decomposition – K-SVD)

Bu bölümde sözlük eğitim yöntemlerinden bir diğeri K adet – Tekil Değer Ayrıştırma (K-Singular Value Decomposition – K-SVD) [2, 3, 11, 20] algoritması tanıtılacaktır. Bu algoritmada da, MOD algoritmasında olduğu gibi katsayıların bulunması aşamasında bölüm (2.1)'de açıklanan herhangi bir vektör seçim algoritmasının kullanılmasına izin verilmektedir. Tez çalışmasında; KSVD algoritmasında da MOD algoritmasında olduğu gibi ağırlık matrisinin oluşturulması sürecinde OMP yönteminden yararlanılmıştır. Sözlük tanılamada kullanılan vektör seçim algoritması, K-ortalamalar yönteminin genelleştirilmiş tasarımı şeklinde düşünülebilir [2].

K-SVD algoritmasını detayları ile açıklamak üzere, denklem (3.16)'da verildiği şekli ile bir nesnel fonksiyon tanımlayabiliriz. Bu fonksiyon önceki bölümlerde seyrek gösterilim kavramını açıklamak üzere verilen denklem (2.2)'de eşdeğer bir ifadeye sahiptir [3].

$$\min_{D,W}\{||X - DW||_F^2\} \qquad \forall i, ||w_i||_0 \leq T_0 \qquad (\mathbf{3.16})$$

Algoritmayı tanımlarken öncelikle seyrek kodlama durumunu araştırmak gerekmektedir, yani W olarak gösterilen katsayılar matrisi ile seyrek gösterilim aranmaktadır. Bu durumda D sözlük matrisi sabit tutulmaktadır. Denklem (3.16) yardımıyla bu aşamada katsayı vektörlerini bulmak üzere denklem (3.17) tanımlanabilir.

$$||X - DW||_F^2 = \sum_{i=1}^{N} ||x_i - Dw_i||_2^2 \qquad (\mathbf{3.17})$$

Denklem (3.16) ile tanımlanan ifadeye N iterasyon sayısının ilave edilmesi ile problem denklem (3.18) ile tanımlanan şekle dönüşecektir.

$$i = 1,2,\dots,N, \qquad \min_{w_i} \left\{ \left|\left| x_i - Dw_i \right|\right|_2^2 \right\} \qquad ||w_i||_0 \leq T_0 \qquad (\mathbf{3.18})$$

İfadelerden de anlaşılacağı üzere problemin bu aşaması bölüm (2.1)'de açıklanan bir Takip Algoritması ("Pursuit Algorithm - PA") kullanılarak çözüme kavuşturulabilir. Ayrıca denklemde yer alan T_0 yeterince küçük olduğu müddetçe bu çözüm ideal olana oldukça yakın bir çözüm olacaktır [3].

Problemin ikinci aşaması sıfırdan farklı olan katsayı elemanları ile sözlük matrisinin güncellenmesi ve hesaplanmasıdır. Bu aşamada W ve D matrislerinin sabit olduğu ve sözlük matrisinin sadece bir sütunu d_i, ile ilgilenildiği söylenebilir. Benzer şekilde bu sütuna karşılık gelecek, katsayı matrisinden i. Satırındaki w_T^i hesaba katılacaktır. Denklem (3.16) ile tanımlanan nesnel fonksiyona geri dönülürse sonuç terim denklem (3.19)'de yazıldığı şekilde ifade edilebilir.

$$\left|\left|X - DW\right|\right|_F^2 = \left\| X - \sum_{j=1}^{K} d_j w_T^j \right\|_F^2 = \left\| \left(X - \sum_{j \neq k} d_j \, w_T^j \right) - d_k w_T^k \right\|_F^2$$
$$= ||E_k - d_k w_T^k||_F^2 \qquad (\mathbf{3.19})$$

Bu ifade ile DW çarpımı, K matrisin toplamı şekline dönüştürülmüştür. Terimde $K-1$ adet terimin sözlük güncellemesi gerçekleştirilmiş ve k. terimin sonucu aranmaktadır [3]. İfade de yer alan E_k matrisi, N örnek için k. terime ulaşıldığındaki hatayı göstermektedir.

Problemin bu aşamasında $d_k$ve w_T^k terimlerinin bulunmasına alternatif olarak Tekil Değer Ayrıştırma (Singular Value Decomposition –SVD) yönetiminin kullanılması önerilebilir. SVD yöntemi E_k terimine yakınsayan rank-1 matrisini bulmaktadır. Bu durum denklem (3.19)'de tanımlanan hatayı en küçük duruma getirmektedir [2]. Bunun yanında bu adım w_T^k rasgele bulunduğu için bazı hatalar içerebilir.

Yukarıdaki problemin bir çözümü olarak, $\{x_i\}$ örneklerini gösteren ve d_k sözlük atomlarını içeren v_k vektörleri tanımlanabilir. Bu tanımlamanın yapıldığı yerlerde

$w_T^k(i)$ katsayı değerleri sıfırdan farklıdır. İfade denklem (3.20) verilen şekli ile gösterilebilir.

$$v_k = \{i |\ 1 \leq i \leq K, w_T^k(i) \neq 0\} \qquad (\mathbf{3.20})$$

Şimdi $(v_k(i), i)$ girişleri bir olan geri kalan elemanları ise sıfır olan $N \times |v_k|$ boyutunda bir matris, Ω_k, tanımlansın. Bu matris ile katsayı matrisinin çarpılması, $w_R^k = w_T^k \Omega_k$, $|v_k|$ uzunluğunda bir satır vektörünün oluşmasını sağlayacaktır [3]. Benzer şekilde aynı matrisin veri kümesi ile çarpılması, $X_k^R = X\Omega_k$, d_k sözlük atomlarını kullanan ve boyutu $n \times |v_k|$ olan bir alt örnek kümesinin oluşmasını sağlayacaktır. Aynı durum d_k sözlük atomlarını kullanan hata vektörünün seçimi içinde uygulanabilir.

Gerçekleştirilen tanımlamalardan sonra d_k ve w_T^k sağlamak üzere denklem (3.19) ile verilen küçültme ifadesine geri dönülebilir. Yeni tanımlama denklem (3.21) ile verilmiştir. Bu sefer, orijinal çözüm ile aynı sonucu verecek, $\check{x}_T^k$, çözümü aranmaktadır.

$$||E_k\Omega_k - d_k w_T^k \Omega_k||_F^2 = ||E_k^R - d_k x_R^k||_F^2 \qquad (\mathbf{3.21})$$

Denklemde verilen çözüm direkt olarak SVD yöntemi ile yapılabilir. Gerçekte, bu ifade denklem (3.19)'de gözüken hatayı basitleştirmekte ve iki uyarlamalı bölümden oluşmaktadır. İlk bölümde k adet atom kullanılarak örnekler oluşturulmakta ikinci bölümde geri kalanı kullanılmaktadır.

SVD algoritması ile sıkıştırılmış matris E_k^R, $E_k^R = U\Delta V^T$ olacak şekilde ayrıştırılabilir. İfadede U matrisinin birinci sütunu $\check{d}_k$ sözlük sütunun çözümü ve V matrisinin birinci sütunu katsayı vektörü, w_R^k, şeklinde tanımlanmaktadır. Bu çözümün gerçeklenmesi için sözlük matrisinin normalize edildiği varsayımının kabul edilmesi gerekir.

SVD çözümüne alternatif olarak, denklem (3.22) ile verilen ifade birkaç iterasyon takip edilerek sonuca ulaşılabilir. Bu ifade de d_k sözlük vektörünün normalleştirilmesi için iki vektör ölçeklenmektedir [3].

$$\check{d}_k = \frac{E_k^R {w_R^k}^T}{w_R^k \cdot {w_R^k}^T}, \qquad w_R^k = \frac{\check{d}_k^T E_k^R}{\check{d}_k^T \check{d}_k} \qquad (\mathbf{3.22})$$

Yukarıda verilen denklem ile sözlük ve katsayı matrislerinin güncellenmesi sağlanmıştır. Bu yöntem k-ortalamalar yöntemine pareler olarak K-SVD ismi ile anılmaktadır.

K-SVD algoritmasında her zaman SVD adımlarından sağlanan en güncel katsayılar kullanılmaktadır [2, 11]. Bu algoritmaya paralel şekilde elde edilen sözlük çözümleri yeni adımda katsayı matrisinin güncellemesi için kullanılabilir.

MOD algoritmasına benzer şekilde K-SVD algoritması da aşamalı şekilde yazılmak istenirse algoritma adımları aşağıdaki şekilde tanımlanacaktır [1]. Benzer şekilde D_0 başlangıç sözlüğü olarak eğitim kümesinin, yani veri kümesinin, ilk K vektörü alınabilir, ancak bu vektörlerin normalize edilmesi gerekmektedir. j iterasyon indisini göstermektedir. i ise veri kümesindeki sütunların indisini ifade etmekte ve $i = 1 \ldots . N$ arasında değer almaktadır.

1. j. iterasyonda, x_i, i. veri vektörünün alınması. X veri matrisi, $X = \{x_i\}_{i=1}^{N}$ şeklinde N adet sütundan oluşmaktadır ve her bir sütun x_i veri vektörü olarak görülebilir.
2. D_{j-1} sözlük matrisini, , x_i , i . veri vektörünü ve bir vektör seçim algoritmasını (bu çalışma için OMP kullanılmıştır) kullanarak w_i katsayı vektörlerinin bulunması. Elde edilen w_i vektörü W katsayı matrisinin i. sütununu göstermektedir. Bu adımdaki işlem j. iterasyonda, tüm x_i değerleri alınarak w_i 'lerin hesaplanmasını içerir. w_i vektörlerinin birleşmesi ile $W = \{w_k\}_{i=1}^{N}$ ağırlık matrisi oluşacaktır.
3. j. iterasyonda denklem (3.19) kullanılarak D_j sözlük matrisinin hesaplanması.
4. j iterasyon değerinin arttırılarak 1. adıma geri dönülmesi.

4. SÖZLÜK EĞİTİMİNDE UYARLAMALI YÖNTEMLER

Tez çalışmasının bu bölümünde geleneksel olarak niteleyebileceğimiz blok güncellemeli sözlük öğrenme yöntemlerine [KSVD, MOD] ilave olarak yakın zamanda sunulmuş olan uyarlamalı bir sözlük öğrenme algoritması [1] açıklanacaktır.

4.1 Uyarlamalı Optimal Yönler Metodu ("Recursive Least Squares Method of Optimal Directions - RLS-DLA")

Bu bölümde güncel bir algoritma olarak Uyarlamalı Optimal Yönler Metodu ("Recursive Least Squares Method of Optimal Directions - RLS-DLA") [1] açıklanacak ve gerçekleme yolları sunulacaktır. Bu algoritmanın en önemli avantajlarından biri her bir eğitim vektörü için sözlük çözümünün tekrar hesaplanmasıdır. Bu nedenle, uyarlamalı yöntemler veri kümesinin tümünü aynı anda kullanan geleneksel blok yöntemlere göre sonuca çok daha hızlı yakınsama sağlamaktadır. Uyarlamalı Optimal Yönler Metodunun (RLS-DLA), bölüm (3.2)'de açıklanan Optimal Yönler Metodu (MOD) üzerine gerçekleneceği söylenebilir.

Algoritmanın temel noktası, ilk i eğitim vektöründen çıkan sözlük matrisinin bilinmesi durumunda, gelecek yeni bir eğitim vektörü için sözlüğün güncellenmesi ya da tekrar hesaplanması şeklindedir. Algoritmada matrisler zaman adımını gösteren alt indisleri ile birlikte ifade edilmektedir. Bu durumda veri matrisi vektörler yardımıyla $X_i = [x_1,\ x_2, \dots, x_i]$ şeklinde gösterilebilir. Bu durumda matrisin boyutu $M \times i$ şeklinde olacaktır. Verilen gösterimde her bir x_i veri kümesindeki bir örneği belirtmektedir ve toplam M adet elemanı vardır. Benzer şekilde katsayılar matrisi boyutu $K \times i$ olmak üzere $W_i = [w_1,\ w_2, \dots, w_i]$ ifadesi yardımıyla gösterilebilir. Bu durumda bölüm (3.2)'de ifade edildiği üzere, Optimal Yönler Metodu (MOD) i eğitim vektörünün indisini göstermek üzere aşağıdaki denklemler yardımıyla ifade edilebilir. Denklemde yer alan D_i, i. vektöre gelindiğinde hesaplanan sözlük matrisi kestirimini göstermektedir ve matrisin boyutu $M \times K$ şeklindedir. RLS-DLA

yöntemi bu denklem temel alınarak ya da diğer bir ifadeyle MOD algoritması temel alınarak gerçeklenecektir [1].

$$D_i = (X_i W_i^T)(W_i W_i^T)^{-1} = B_i C_i \qquad (\mathbf{4.1})$$

Denklem (4.1)'de yer alan B_i ve C_i terimleri aşağıda yer alan denklem (4.2) ve denklem (4.3) yardımıyla daha açık şekilde ifade edilebilir.

$$B_i = (X_i W_i^T) = \sum_{j=1}^{i} x_j w_j^T \qquad (\mathbf{4.2})$$

$$C_i^{-1} = (W_i W_i^T) = \sum_{j=1}^{i} w_j w_j^T \qquad (\mathbf{4.3})$$

Hesaplamanın şimdiki aşamasında yeni bir eğitim vektörü, $x = x_{i+1}$, geldiğini göz önüne alınabilir. Notasyonda meydana gelebilecek karışıklığı önlemek için vektörler için zaman göstergesi yani zaman alt indisi bu bölümde göz ardı edilmiştir. Bu durumda i. adımdaki vektöre karşılık gelebilecek katsayı (ağırlık) vektörü $w = w_{i+1}$ şeklinde gösterilebilir ve D_i sözlüğü yardımıyla uygun bir vektör seçim algoritması kullanılarak hesaplanır. Bu çalışmada çalışmanın diğer bölümlerinde de belirtildiği üzere vektör seçim algoritması olarak dik seçim algoritması (OMP) [9, 10] kullanılmıştır. Belirtilen varsayımlar altında i. adımda hesaplanan veri vektörü ve ortaya çıkacak hata denklem (4.4) ve denklem (4.5) yardımıyla belirtilebilir.

$$\tilde{x} = D_i w = B_i C_i w \qquad (\mathbf{4.4})$$

$$r = x - \tilde{x} \qquad (\mathbf{4.5})$$

Sırada yer alan eğitim vektörü hesaba katılarak yeni sözlük vektörü $D_{i+1} = B_{i+1} C_{i+1}$ şeklinde ifade edilir ve hesaplanır. Bu ifadede yer alan B_{i+1} ve C_{i+1} terimleri;

$$B_{i+1} = B_i + x w^T \qquad (\mathbf{4.6})$$

$$C_{i+1}^{-1} = C_i^{-1} + w w^T \qquad (\mathbf{4.7})$$

denklemlerinde verildiği gibi bulunur. C_{i+1} terimi, matris tersi lemması kullanılarak denklem (4.8) de verilen ifade yardımıyla hesaplanabilir [1].

$$C_{i+1} = C_i - \frac{C_i w w^T C_i}{w^T C_i w + 1} \qquad (\mathbf{4.8})$$

Benzer şekilde tüm bu terimler yardımıyla D_{i+1} sözlüğü denklem (4.9)'da verildiği şekli ile bulunabilir. Bu denklem yeni eklenen eğitim kümesindeki terim yardımıyla hesaplanan sözlük matrisini göstermektedir.

$$D_{i+1} = (B_i + xw^T)\left(C_i - \frac{C_i w w^T C_i}{w^T C_i w + 1}\right)$$

$$= B_i C_i - B_i \frac{C_i w w^T C_i}{w^T C_i w + 1} + x w^T C_i - x w^T \frac{C_i w w^T C_i}{w^T C_i w + 1} \qquad (\mathbf{4.9})$$

Tüm bu denklemlerde yer alan C_i matrisi simetrik bir matristir. Bu durumda $C_i = C_i^T$ şeklinde yazmak mümkündür. Benzer şekilde denklem (4.7)'de yer verilen C_i^{-1} matrisi de simetrik bir matristir.

Algoritmanın daha basit şekilde ifade edilmesini sağlamak üzere denklem (4.10) ve denklem (4.11)'de yer alan u vektörü ve skaler α tanımlanabilir.

$$u = C_i w, \qquad u^T = w^T C_i \qquad (\mathbf{4.10})$$

$$\alpha = \frac{1}{1 + w^T C_i w} = \frac{1}{1 + w^T u} \qquad (\mathbf{4.11})$$

$$1 - \alpha = \frac{w^T u}{1 + w^T u} = \frac{w^T C_i w}{1 + w^T C_i w}$$

Yeni tanımlardan yaralanarak sözlük matrisinin güncellenmesi aşağıda ifade edilen denklem (4.12) ve denklem (4.13)'deki gibi yapılabilir. Denklem (4.13), RLS-DLA yöntemi kullanılarak sözlük matrisinin güncellenmesine ait ifadeyi vermektedir.

$$C_{i+1} = C_i - \alpha u u^T \qquad (\mathbf{4.12})$$

$$D_{i+1} = D_i - \alpha \tilde{x} u^T + x u^T - (1 - \alpha) x u^T = D_i + \alpha r u^T \qquad (\mathbf{4.13})$$

Verilen bu denklemler yardımıyla en basit şekilde Uyarlamalı Optimal Yönler Metodu (RLS-DLA) tanımlanmıştır [1]. Algoritmanın tüm bu aşamalarında matrislerin toplanması ve matris vektör çarpımları şeklinde basit matematiksel işlemler yapılmaktadır. Hatta gerçekleştirilen çarpma işlemlerinden bazılarının seyrek matrisler ile yapılması işlemlerin daha da kolay olmasını sağlamaktadır. Diğer algoritmalarda olduğu şekilde matris çarpımlarının ve matris tersinin alınması işlemlerinin bu algoritmada olmaması işlemsel karmaşıklığı azaltmaktadır.

Katsayı (ağırlık) vektörünün, w_i , seyrek bir vektör olduğu yani sadece belirli sayıda elemanının sıfırdan farklı olduğu daha önce söylenmişti. Ağırlık vektörü için

seyrekliğin en uç değeri bir eleman dışında tüm değerlerin sıfır olmasıdır. Geri kalan bu tek elemanın değeri ise 1 olarak verilebilir. Bu anlamda Uyarlamalı Optimal Yönler Metodu (RLS-DLA), sürekli k-ortalama yöntemine benzerlik göstermektedir.

Uyarlamalı Optimal Yönler Metodu (RLS-DLA) yöntemi ile sözlük matrisinin güncellenmesi diğer bir şekilde aşağıda yer alan denklem (4.14) ile yapılabilir. Bu denklem daha önce verilen sözlük güncelleme ifadesinin eşdeğeridir.

$$D_{i+1} = D_i + rw^T(\alpha C_i) = D_i + \frac{(rw^T)C_i}{1 + w^T C_i w} \qquad (\mathbf{4.14})$$

Bu denklem özellikle seyreklik değerinin uç noktalarda olduğu, örneğin bir eleman dışında tüm diğerlerin sıfır olması, durumlarda kullanılması daha uygun olabilir. Denklemlerde verilen C_i matrisi aynı zamanda diyagonal bir matristir.

Çalışmanın bu bölümüne kadar açıklanan RLS-DLA algoritmasında, uyarlamalı yöntemlerde genellikle mevcut olan unutma faktörü λ ya yer verilmemiştir. Tüm bu ifadelerin üzerine RLS-DLA algoritmasının türevi düşünülerek diğer uyarlamalı algoritmalarda olduğu gibi unutma faktörü ilave edilebilir [1, 21]. Bu durumda ilk i eğitim vektörü için sözlük; bir bedel fonksiyonunu minimize edecek şekilde hesaplanır. Bedel fonksiyonunun en genel hali denklem (4.15)'de verildiği şekildedir.

$$argmin_{D,W} f(D,W) = argmin_{D,W} \sum_i \left|\left|r_i\right|\right|_2^2$$

$$= argmin_{D,W} \sum_i ||X - DW||_F^2 \qquad (\mathbf{4.15})$$

Unutma faktörünün λ olduğu göz önüne alınırsa bedel fonksiyonu, en az kareler hatasının toplamını kullanarak denklem (4.16)'da verildiği şekli ile ifade edilebilir. Denklemde yer alan unutma faktörü, $0 < \lambda \leq 1$, arasında değer almakta ve çeşitli fonksiyonlar yardımıyla ifade edilebilmektedir. Unutma faktörü ile ilgili daha detaylı açıklamalar bu bölümde bir alt başlı şeklinde açıklanacaktır.

$$f(D) = \sum_{j=1}^{i} \lambda^{i-j} \left|\left|r_j\right|\right|_2^2 \qquad (\mathbf{4.16})$$

Bölümün bu aşamasında unutma faktörü göz önüne alınarak sözlük güncelleme için gerekli olan tüm denklemler tekrar çıkartılarak RLS-DLA algoritması

oluşturulacaktır. Daha önce tanımlandığı üzere, yeni bir eğitim kümesi vektörünün x, ve buna ait katsayı vektörünün, w, vektörünün mevcut olduğunu düşünelim. Bu durumda güncellenen sözlük matrisi denklem (4.17)'de ifade edildiği şekilde olacaktır.

$$\begin{aligned} D_{i+1} &= argmin_D f(D) \\ &= argmin_D \sum_{j=1}^{i+1} \lambda^{i+1-j} \left|\left|r_j\right|\right|_2^2 \\ &= argmin_D \left(\lambda \left|\left|\widehat{X_\iota} - D\widehat{W_\iota}\right|\right|_F^2 + \left|\left|x - Dw\right|\right|_2^2\right) \end{aligned} \quad (\mathbf{4.17})$$

Denklemlerde yer alan $\widehat{X_\iota}$ ve $\widehat{W_\iota}$ matrisleri önceki aşamada açıklanan matrislerin ölçeklenmiş versiyonlarıdır. Bu matrisler denklem (4.18) ve denklem (4.19)'da verildiği şekilde yinelemeli olarak ifade edilebilir.

$$\widehat{X_\iota} = \left[\sqrt{\lambda}\widehat{X_{\iota-1}}, x_i\right], \quad \widehat{X_1} = x_1 \quad (\mathbf{4.18})$$

$$\widehat{W_\iota} = \left[\sqrt{\lambda}\widehat{W_{\iota-1}}, w_i\right], \quad \widehat{W_1} = w_1 \quad (\mathbf{4.19})$$

Önceki algoritmalara benzer şekilde unutma faktörleri göz önüne alındığında da sözlük matrisi B_{i+1} ve C_{i+1} yardımıyla yazılabilir. Ancak farklı olarak algoritmanın bu kısmında $\widehat{X_\iota}$ ve $\widehat{W_\iota}$ matrisleri kullanılacaktır. B_{i+1} ve C_{i+1} matrisleri yinelemeli şekilde denklem (4.20) ve denklem (4.21)'de verilen şekilde gösterilebilir [1].

$$\begin{aligned} B_{i+1} &= \hat{X}_{i+1}\widehat{W}_{i+1}^T = \left[\sqrt{\lambda}\hat{X}_i, x\right]\left[\sqrt{\lambda}\widehat{W}_i, w\right]^T = \lambda\hat{X}_i\widehat{W}_i^T + xw^T \\ &= \lambda B_i + xw^T \end{aligned} \quad (\mathbf{4.20})$$

$$\begin{aligned} C_{i+1}^{-1} &= \widehat{W}_{i+1}\widehat{W}_{i+1}^T = \left[\sqrt{\lambda}\widehat{W}_i, w\right]\left[\sqrt{\lambda}\widehat{W}_i, w\right]^T = \lambda\widehat{W}_i\widehat{W}_i^T + ww^T \\ &= \lambda C_i^{-1} + ww^T \end{aligned} \quad (\mathbf{4.21})$$

C_{i+1} terimi, önceki algoritmalarda olduğu gibi matris tersi lemması kullanılarak denklem (4.22)'de verilen ifade yardımıyla hesaplanabilir.

$$C_{i+1} = \lambda^{-1}C_i - \frac{\lambda^{-1}C_i w w^T \lambda^{-1} C_i}{w^T\lambda^{-1}C_i w + 1} \quad (\mathbf{4.22})$$

Tüm bu denklemlerden sözlük güncelleme ifadesi denklem (4.23)'de olduğu şekilde çıkacaktır.

$$D_{i+1} = B_{i+1}C_{i+1} = (\lambda B_i + xw^T).\left(\lambda^{-1}C_i - \frac{\lambda^{-1}C_i w w^T \lambda^{-1} C_i}{w^T \lambda^{-1} C_i w + 1}\right)$$

$$= B_i C_i - \lambda B_i \frac{\lambda^{-1}C_i w w^T \lambda^{-1} C_i}{w^T \lambda^{-1} C_i w + 1} + x w^T \lambda^{-1} C_i$$

$$- x w^T \frac{\lambda^{-1}C_i w w^T \lambda^{-1} C_i}{w^T \lambda^{-1} C_i w + 1} \qquad (\mathbf{4.23})$$

Problemin bu aşamasında u vektörü ve skaler α değerini tanımlamak mümkündür. Gerekli denklemler aşağıda denklem (4.24), denklem (4.25a) ve denklem (4.25b) ile verilmiştir.

$$u = (\lambda^{-1}C_i)w, \qquad u^T = w^T(\lambda^{-1}C_i) \qquad (\mathbf{4.24})$$

$$\alpha = \frac{1}{1 + w^T(\lambda^{-1}C_i)w} = \frac{1}{1 + w^T u} \qquad (\mathbf{4.25}\boldsymbol{a})$$

$$1 - \alpha = \frac{w^T u}{1 + w^T u} = \frac{w^T(\lambda^{-1}C_i)w}{1 + w^T(\lambda^{-1}C_i)w} \qquad (\mathbf{4.25}\boldsymbol{b})$$

Sözlük matrisinin güncellemesi u vektörü ve skaler α değeri yardımıyla denklem (4.26) ve denklem (4.27) yardımıyla ifade edilebilir.

$$C_{i+1} = (\lambda^{-1}C_i) - \alpha u u^T \qquad (\mathbf{4.26})$$

$$D_{i+1} = D_i + \alpha r u^T \qquad (\mathbf{4.27})$$

Son aşamada elde edilen sözlük güncelleme ifadesi, unutma faktörü ilave edilmediği durumdaki denklemlere, denklem (4.12) ve denklem (4.13), oldukça benzerdir. Aradaki tek fark denklemlerden de görüleceği üzere C_i yerine $(\lambda^{-1}C_i)$ ifadesinin kullanılmasıdır.

D_0 bir başlangıç sözlüğünü ve C_0 bir başlangıç C matrisini göstermek üzere Uyarlamalı Optimal Yönler Metodu (RLS-DLA) aşamalı şekilde aşağıdaki şekilde de ifade edilebilir [1]. C_0 başlangıç matrisi genellikle bir birim matris olarak düşünülebilir. Algoritma adımlarında yer alan unutma faktörü $0 \ll \lambda_i \leq 1$ arasında değer almaktadır. İterasyon alt indis i ile gösterilmektedir.

RLS-DLA sözlük öğrenme algoritması:

1. Yeni bir eğitim kümesi vektörünün alınması, x_i
2. D_{i-1} sözlük matrisini ve bir vektör seçim algoritmasını (bu çalışma için OMP kullanılmıştır) kullanarak w_i katsayı vektörlerinin bulunması

3. Gösterim hatasının (kestirim hatasının) bulunması, $r = x_i - D_{i-1}w_i$
4. Unutma faktörünün uygulanması, $C_{i-1}^{*} = \lambda_i^{-1}C_{i-1}$
5. u vektörünün hesaplanması, $u = C_{i-1}^{*}w_i$
6. Skaler α değerinin hesaplanması, $\alpha = 1/(1 + w_i^T u)$
7. Sözlük matrisinin güncellemesi, $D_i = D_{i-1} + \alpha r u^T$
8. Gelecek iterasyon için C matrisinin güncellenmesi, $C_i = C_{i-1}^{*} - \alpha u u^T$
9. Hesaplanan sözlüğün normalize edilmesi.

D_0 başlangıç sözlüğü olarak eğitim kümesinin ilk K vektörü alınabilir, ancak bu vektörlerin normalize edilmesi gerekmektedir. Denklem (4.7)'den de anlaşılacağı üzere C_0 matrisi bir diyagonal matristir.

4.1.1 RLS-DLA algoritmasında unutma faktörü

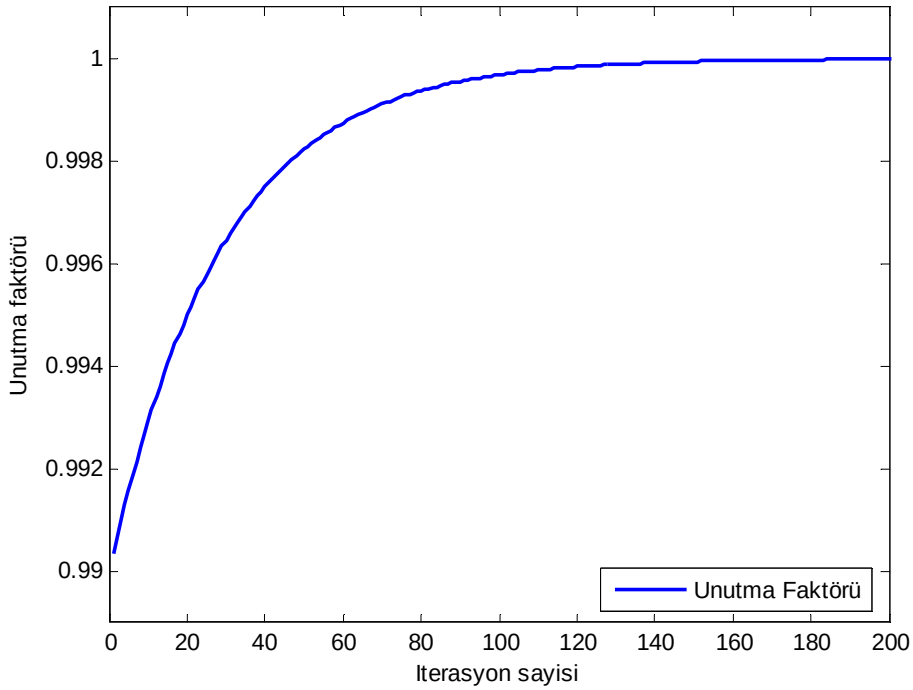
Uyarlamalı Optimal Yönler Metodu (RLS-DLA) gerçeklenirken bir unutma faktörü iterasyon buyunca kullanılabilir [1, 21]. Unutma faktörünün gerçeklenmesi için çeşitli fonksiyonlar mevcuttur. Ancak hepsinde belirli bir λ_0 başlangıç değerinden iterasyon boyunca 1 değerine bir yakınsama gerçekleştirildiği söylenebilir [21]. Kullanılabilecek fonksiyonlardan biri denklem (4.28) ile verilen lineer, karesel ya da kübik fonksiyondur. Fonksiyonda yer alan p parametresi fonksiyonun derecesini göstermektedir. Eğer p 1 ise lineer bir fonksiyon, 2 ise karesel bir fonksiyon, 3 ise kübik bir fonksiyondur. a ise hangi iterasyon değerinden sonra unutma faktörünün 1 değerini alacağını göstermektedir.

$$\lambda_i = \begin{cases} 1 - (1 - \lambda_0)(1 - i/a)^p & eğer\ i \leq a \\ 1 & eğer\ i > a \end{cases} \qquad (\mathbf{4.28})$$

Bunun yanında bu çalışma kapsamında unutma faktörü için denklem (4.29)'da verilen eksponansiyel fonksiyon kullanılmıştır. a yine hangi iterasyon değerinden sonra unutma faktörünün 1 değerini alacağını göstermektedir. i, iterasyon sayısını ifade etmektedir.

$$\lambda_i = 1 - (1 - \lambda_0)\left(\frac{1}{2}\right)^{\frac{i}{a}} \qquad (\mathbf{4.29})$$

Yüksek lisans tez çalışmasında kullanılan unutma faktörünün ***i*** iterasyon değeri ile değişimi Şekil 4.1'de verildiği gibidir. Belirli bir iterasyon sonunda unutma faktörü değerinin 1 olduğu şekilden görülmektedir.



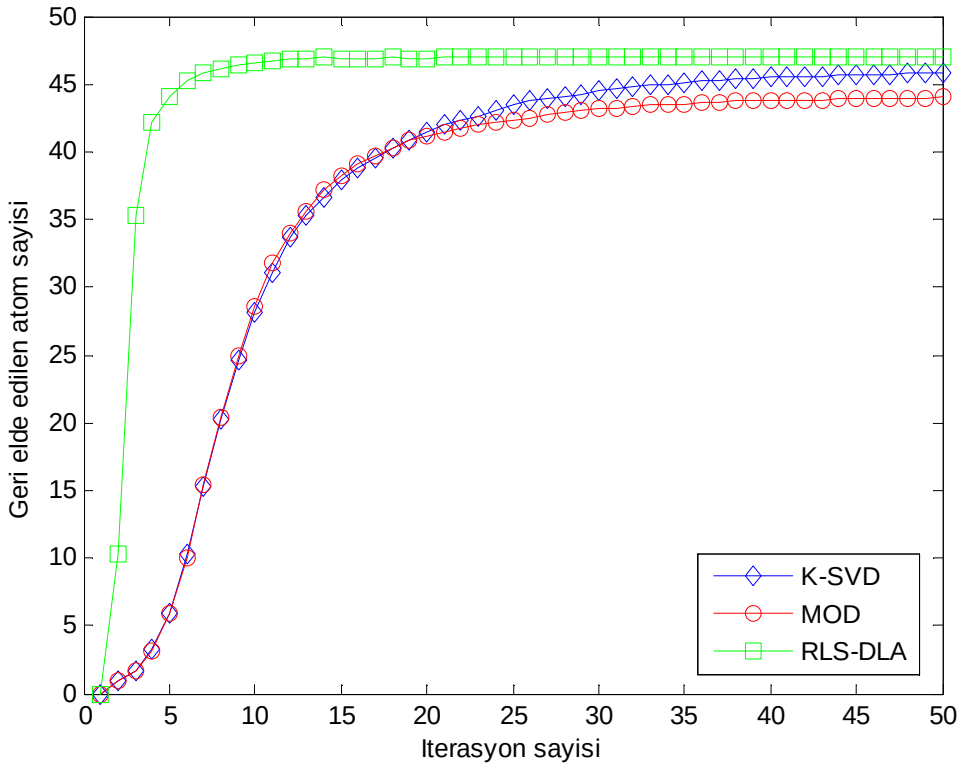
Şekil 4.1 : *i* iterasyon değerine göre unutma faktörü değerinin değişimi.

5. DENEYSEL SONUÇLAR VE UYGULAMALAR

Tez çalışmasının bu bölümde seyrek işaretlerin gösterilimi için önceki bölümlerde tanımlanan MOD, K-SVD ve RLS-DLA sözlük öğrenme algoritmaları gerçeklenerek başarımları çeşitli şartlar altında karşılaştırılacaktır. Tüm gerçeklemelerde normal dağılımlı rasgele süreçlerden öncelikle sözlük matrisi ardından veri matrisi oluşturularak sentetik veriler kullanılmıştır.

Karşılaştırma işlemi sırasında rasgele dağılımlı süreçlerden elde edilen sözlük matrisleri ile belirtilen algoritmalar çalıştırıldıktan sonra oluşan sözlük matrisleri atom atom (sütun bazında) karşılaştırılmıştır. Karşılaştırma sonucunda eş bulunan atomlar (sütunlar) işaretlenerek algoritma sonucunda geri elde edilen atom sayısının oluşmasını sağlamıştır. Sözlük matrisinin boyutunun değiştiği durumlarda ise atom sayısı yerine geri elde edilen atom oranı kavramının kullanıldığı söylenebilir. Geri elde edilen atom oranı, başarılı şekilde geri elde edilmiş atom sayısı ile toplam atom sayısının birbirine oranıdır.

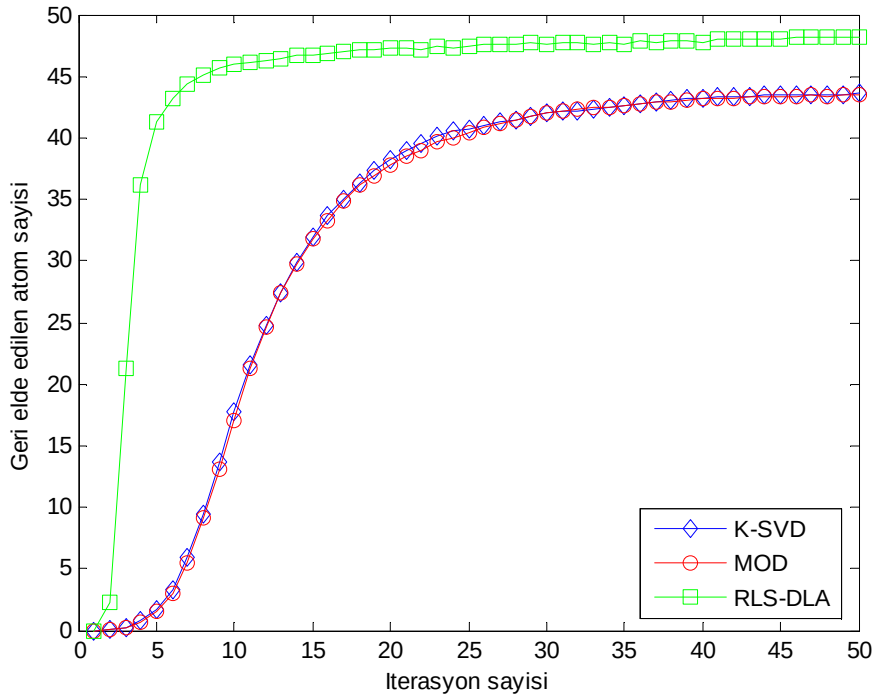
Yapılacak ilk gözlem farklı işaret gürültü oranları (SNR) için MOD, K-SVD ve RLS-DLA algoritmalarının başarımlarının karşılaştırılmasıdır. Burada ifade edilen işaret gürültü oranı sentetik veriye ilave edilen gürültü ile veri arasında oluşmaktadır. Bu gözlemde seyreklik değeri 3, sözlük matrisinin sütun sayısı 50, iterasyon sayısı 50, veri uzunluğu 20 ve veri matrisindeki örnek sayısı 1500 olarak alınmıştır. İlk örnek işaret gürültü oranının 20 dB olduğu durum için verilmiştir. Verilen şartlar altında algoritmaların başarımları yani iterasyon değişimine göre elde edilen sözlük atom sayıları Şekil 5.1'de verilmiştir. Şekil 5.1'dende görüldüğü üzere RLS-DLA yöntemi diğer algoritmalara kıyasla sonuca çok hızlı bir yakınsama sağlamaktadır. İlk 5 iterasyon sonucunda işaretin seyrek gösterilimini sağlayacak sözlük matrisinin büyük bölümü elde edilmiştir. Şekil 5.1'de de görüldüğü gibi ilk 5 iterasyon sonucunda 50 adet atomdan ~43 adet atomun geri elde edildiği söylenebilir. Bu durum 50 iterasyon sonucunda yaklaşık %86 atomun geri elde elde edildiğini göstermektedir.



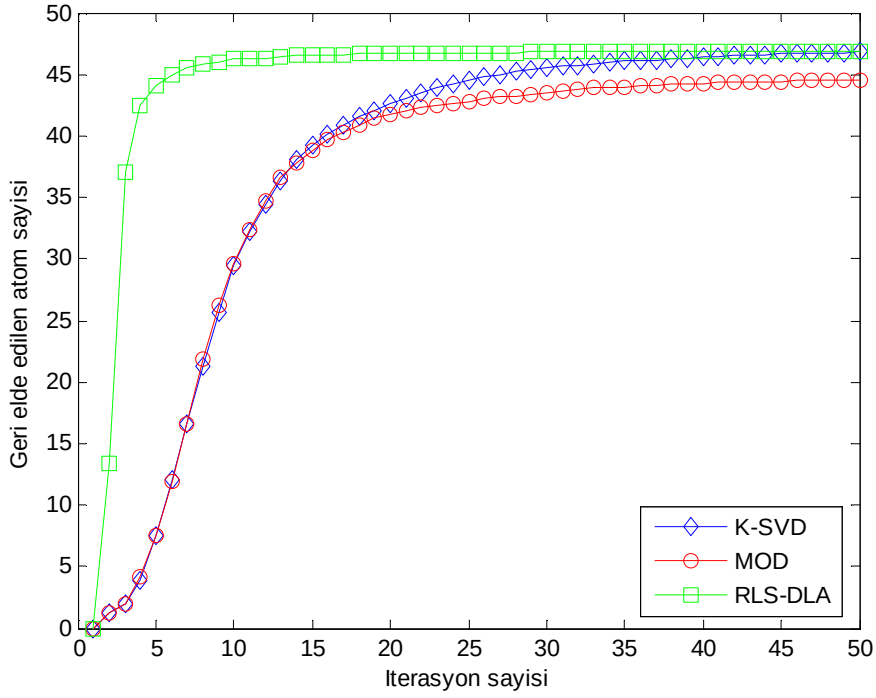
Şekil 5.1 : Veri matrisindeki örnek sayısı 1500, işaret gürültü oranı 20 dB, seyreklik değeri 3, sözlük matrisinin sütun sayısı 50, iterasyon sayısı 50 ve veri uzunluğu 20 şartlarında algoritmaların başarımlarının karşılaştırılması.

Sözlük ve veri matrisi için diğer değişkenler bir önceki gözlem ile aynı kaldığı düşünüldüğünde işaret gürültü oranının 10 dB olarak ayarlandığı deney için elde edilecek başarım Şekil 5.2 ile verilmiştir. Başarımlar açısından çıkan sonuçlar 20 dB işaret gürültü oranı değeri ile benzerdir. Benzer şekilde bu işaret gürültü oranı değeri içinde RLS-DLA algoritması sözlük matrisinin oluşturulmasında diğer yöntemlere göre oldukça başarılıdır. Şekilden de görüleceği gibi; RLS-DLA algoritması diğer iki yönteme göre hem daha hızlı bir yakınsama sağlamakta hem de daha başarılı sonuçlarını çıkmasını sağlamaktadır.

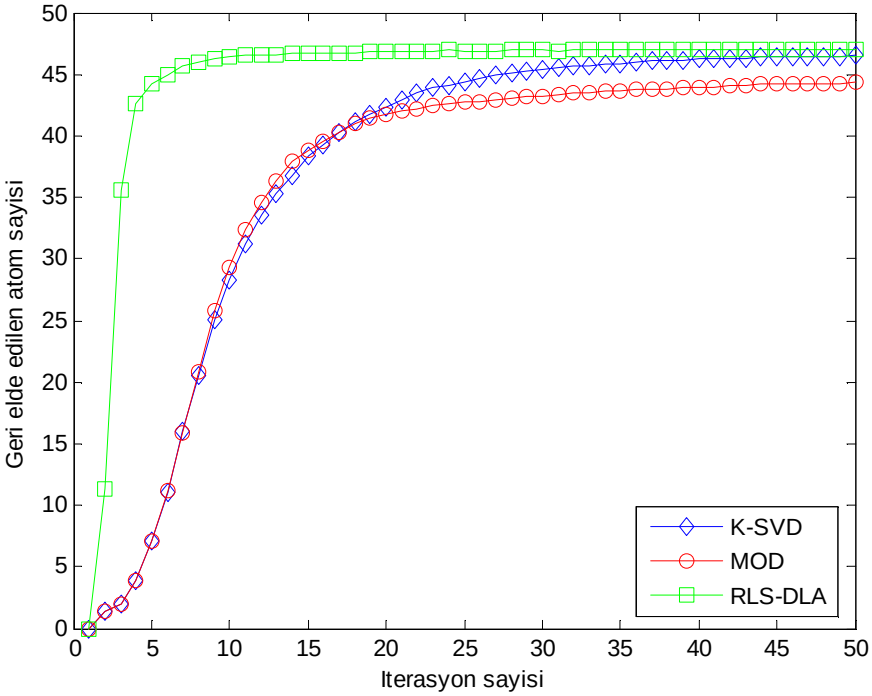
Benzer şekilde işaret gürültü oranının 0 dB, 30 dB ve 60 dB olduğu durumlar içinde sonuçlar aşağıdaki şekillerde sıralı durumda verilmiştir. Şekil 5.3, Şekil 5.4 ve Şekil 5.5'de yer alan sonuçlar da incelendiğinde RLS-DLA algoritmasının diğer algoritmalar ile karşılaştırıldığında daha başarılı sonuçlar verdiği görülecektir.



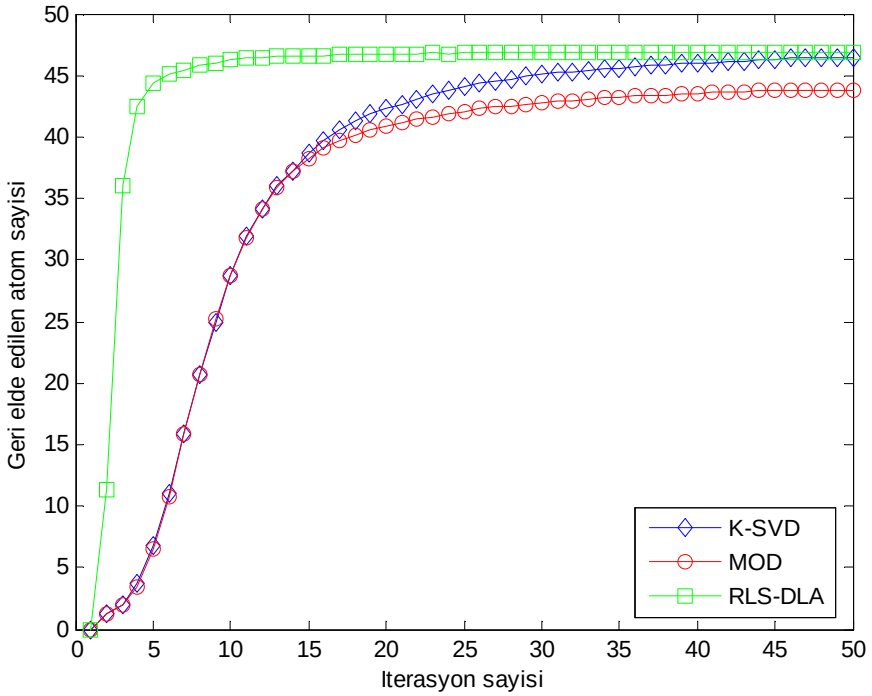
Şekil 5.2 : Veri matrisindeki örnek sayısı 1500, işaret gürültü oranı 10 dB, seyreklik değeri 3, sözlük matrisinin sütun sayısı 50, iterasyon sayısı 50 ve veri uzunluğu 20 şartlarında algoritmaların başarımlarının karşılaştırılması.



Şekil 5.3 : Veri matrisindeki örnek sayısı 1500, işaret gürültü oranı 0 dB, seyreklik değeri 3, sözlük matrisinin sütun sayısı 50, iterasyon sayısı 50 ve veri uzunluğu 20 şartlarında algoritmaların başarımlarının karşılaştırılması.



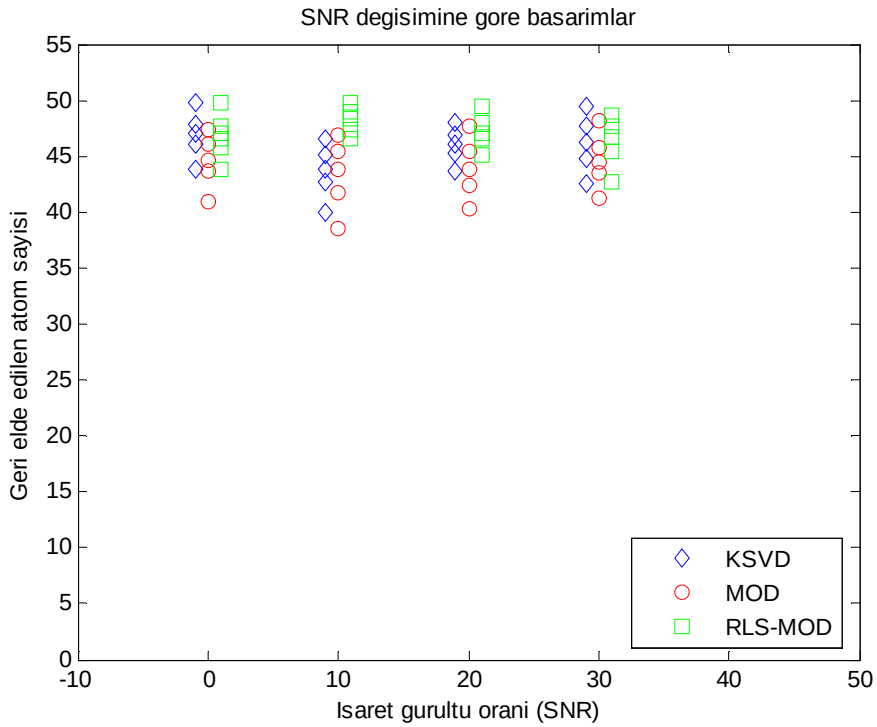
Şekil 5.4 : Veri matrisindeki örnek sayısı 1500, işaret gürültü oranı 30 dB, seyreklik değeri 3, sözlük matrisinin sütun sayısı 50, iterasyon sayısı 50 ve veri uzunluğu 20 şartlarında algoritmaların başarımlarının karşılaştırılması.



Şekil 5.5 : Veri matrisindeki örnek sayısı 1500, işaret gürültü oranı 60 dB, seyreklik değeri 3, sözlük matrisinin sütun sayısı 50, iterasyon sayısı 50 ve veri uzunluğu 20 şartlarında algoritmaların başarımlarının karşılaştırılması.

Çeşitli işaret gürültü oranları için yapılan tüm deneylerde şekillerden de görüleceği üzere RLS-DLA algoritması kullanılan blok yöntemlere göre çok daha hızlı sonuca yaklaşılmasını sağlamaktadır. Ayrıca iterasyon sonlarında elde edilen sözlük atom sayıları da daha yüksek değerlerde olmaktadır.

Çalışmanın bu bölümünde çeşitli işaret gürültü oranları (SNR) için algoritmaların başarımını gözlemlemek üzere değişik bir gözleme başvuralım. Bu gözlemde her bir işaret gürültü oranında toplam 50 adet örnek seti için algoritmalar çalıştırılışmış ve sonuç durumunda geri elde ettikleri atom sayıları 10'lı gruplara ayrılarak ortalamaları alınmıştır. Elde edilen grafik aşağıda Şekil 5.6'da verilmiştir. Şekilden de görüldüğü üzere RLS-DLA algoritması diğer yöntemlere göre daha iyi bir başarım sağlamaktadır.



Şekil 5.6 : Çeşitli işaret gürültü oranları için algoritmaların başarımı.

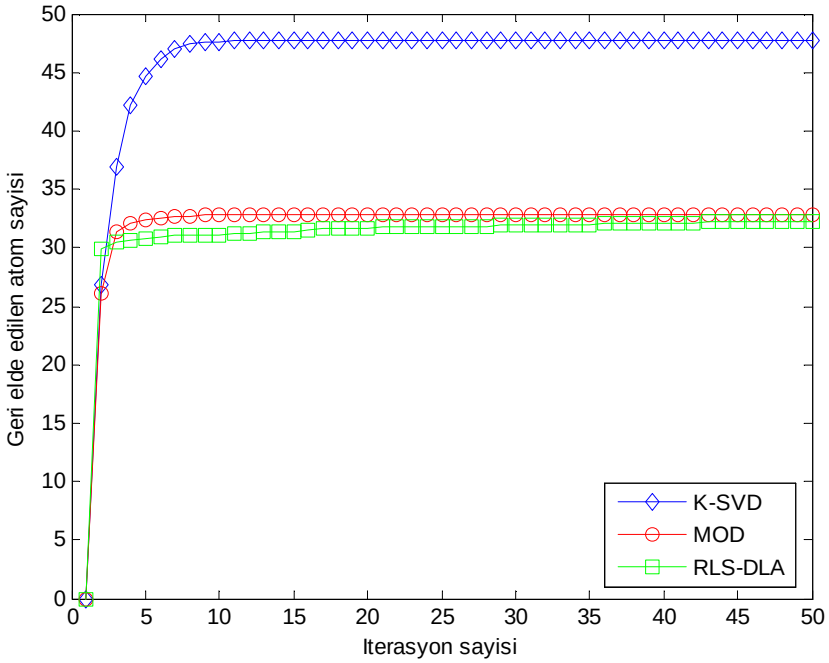
Bu gözlemde elde edilen sonuçlar bir çizelge yardımıyla değersel olarak da ifade edilebilir. (Çizelge 5.1) Elde edilen bu değerler 50 sütunlu bir sözlük matrisinin 50 iterasyon sonunda kaç atomunun geri elde edildiğini göstermektedir.

Çizelgede ifade edilen her bir grup içinde 10 adet örnek vardır. Yani Grup 1 içinde 10 adet örneğin ortalaması alınmıştır.

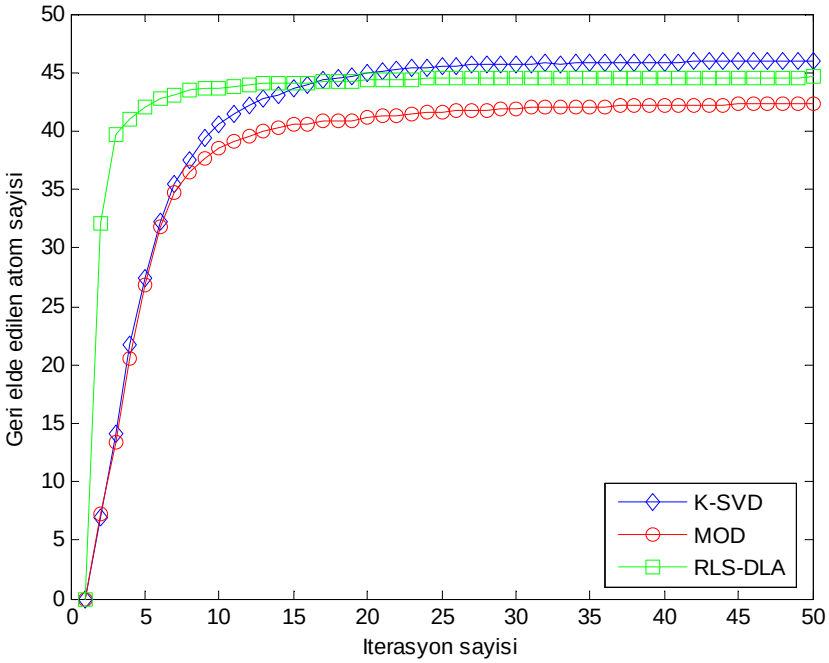
Çizelge 5.1 : Çeşitli işaret gürültü oranları değerlerinde algoritmaların başarımı.

Algoritma	SNR değeri (dB)	Grup 1	Grup 2	Grup 3	Grup 4	Grup 5
MOD	0	40.90	43.60	44.60	46.00	47.40
	10	38.50	41.80	43.80	45.40	46.90
	20	40.30	42.40	43.80	45.50	47.70
	30	41.20	43.50	44.40	45.70	48.10
KSVD	0	43.90	46.10	47.00	47.80	49.80
	10	39.90	42.70	43.90	45.10	46.50
	20	43.60	45.20	46.00	46.90	48.00
	30	42.50	44.80	46.30	47.70	49.40
RLS-DLA	0	43.80	45.70	47.00	47.70	49.80
	10	46.50	47.40	48.30	49.00	49.70
	20	45.10	46.50	47.10	48.00	49.4
	30	42.70	45.50	46.70	47.70	48.70

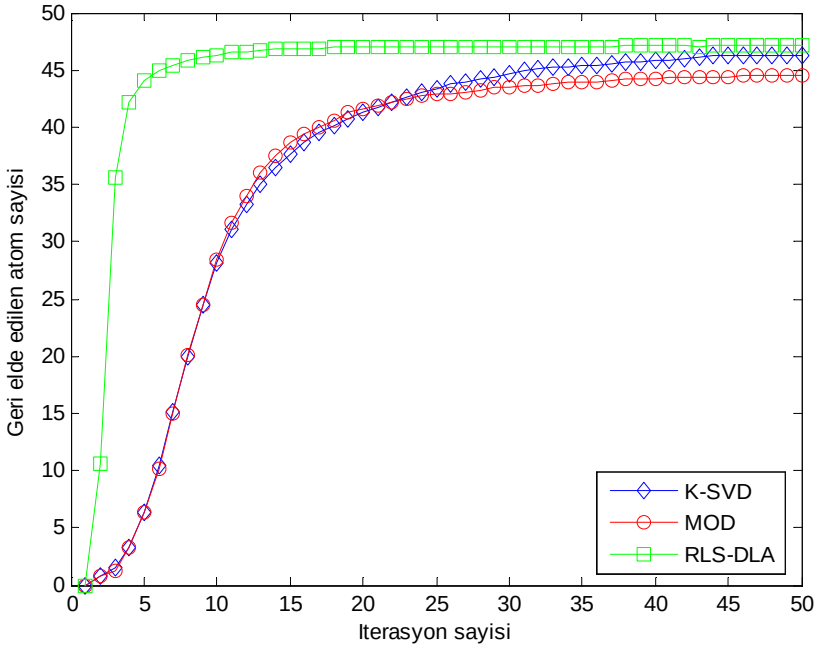
Çalışmanın bu bölümünde diğer bir gözlem N veri matrisindeki örnek sayısı 1500 sabit olmak üzere değişik seyreklik değerleri için başarımların incelenmesine yer verilmiştir. Bu inceleme işaret gürültü oranı 20 dB ve sözlük matrisinin sütun boyutu 50 alınıp gerçekleştirilmiştir. Burada seyreklik değeri aynı zamanda ağırlık (katsayı) matrisinde kaç elemanın sıfırdan farklı olduğunu göstermektedir. Çok düşük seyreklik değeri, örneğin s=1 için, KSVD performansı daha iyi olabilmektedir. Ancak 2 ve üzerindeki tüm s değerleri için RLS-DLA performansı diğer yöntemler daha başarılıdır. 7 ve üzerindeki seyreklik değerlerinin hiç birinde 50 iterasyon sonunda sözlük matrisinin elemanları doğru bir şekilde elde edilememektedir. Bunun nedeni veri kümesinin uzunluğunun 20 olmasıdır. 20 elemanlı bir veri kümesi için ancak 7 seyreklik değerine kadar işlem yapmak mümkün olmaktadır. Şekil 5.7, 5.8, 5.9, 5.10, 5.11 ve 5.12'de seyreklik değeri değişimine göre algoritmaların performansları gösterilmiştir.



Şekil 5.7 : Veri matrisindeki örnek sayısı 1500, seyreklik değeri 1, işaret gürültü oranı 20 dB, sözlük matrisinin sütun sayısı 50, iterasyon sayısı 50 ve veri uzunluğu 20 şartlarında algoritmaların başarımlarının karşılaştırılması.

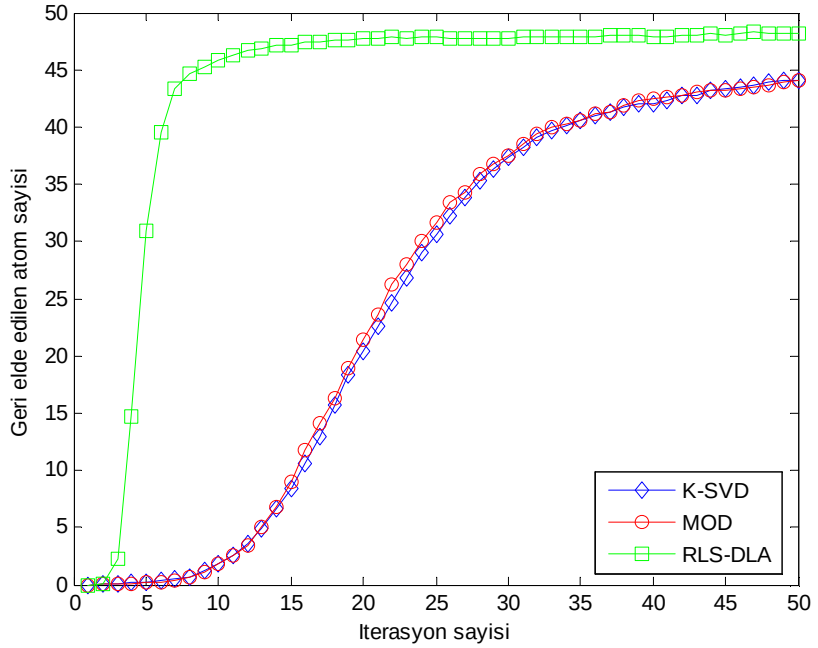


Şekil 5.8 : Veri matrisindeki örnek sayısı 1500, seyreklik değeri 2, işaret gürültü oranı 20 dB, sözlük matrisinin sütun sayısı 50, iterasyon sayısı 50 ve veri uzunluğu 20 şartlarında algoritmaların başarımlarının karşılaştırılması.

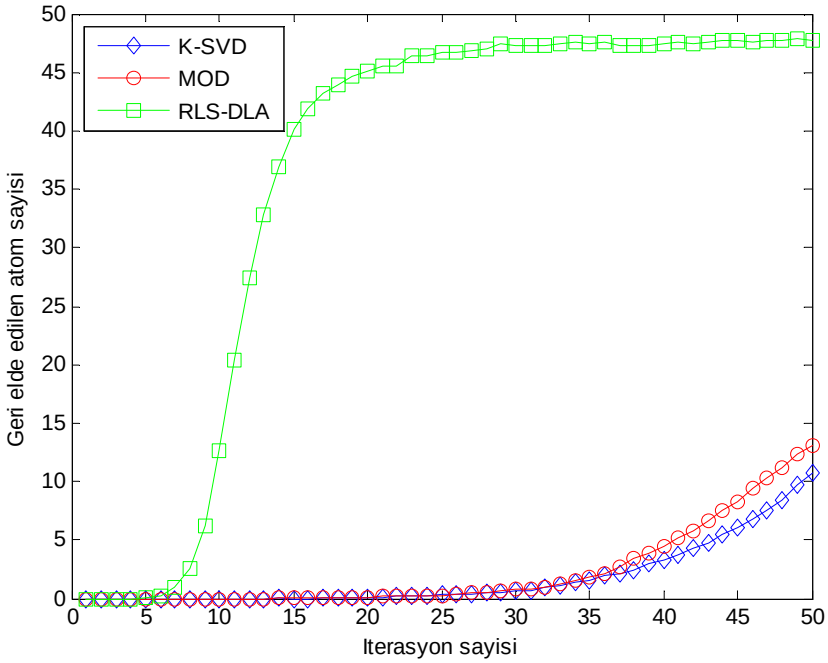


Şekil 5.9 : Veri matrisindeki örnek sayısı 1500, seyreklik değeri 3, işaret gürültü oranı 20 dB, sözlük matrisinin sütun sayısı 50, iterasyon sayısı 50 ve veri uzunluğu 20 şartlarında algoritmaların başarımlarının karşılaştırılması.

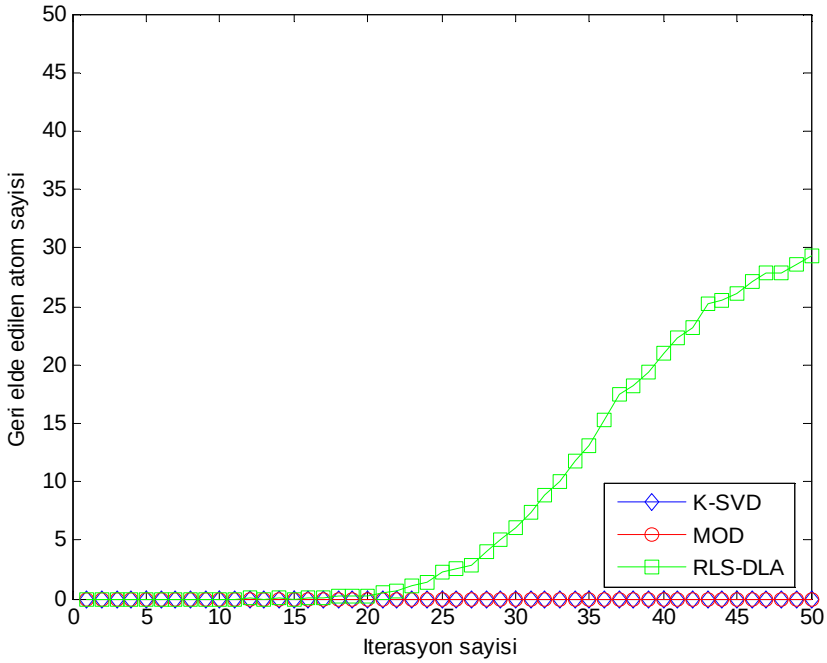
Seyreklik değeri 4 için elde edilen sonuçlar Şekil 5.10 ile verilmiştir.



Şekil 5.10 : Veri matrisindeki örnek sayısı 1500, seyreklik değeri 4, işaret gürültü oranı 20 dB, sözlük matrisinin sütun sayısı 50, iterasyon sayısı 50 ve veri uzunluğu 20 şartlarında algoritmaların başarımlarının karşılaştırılması.

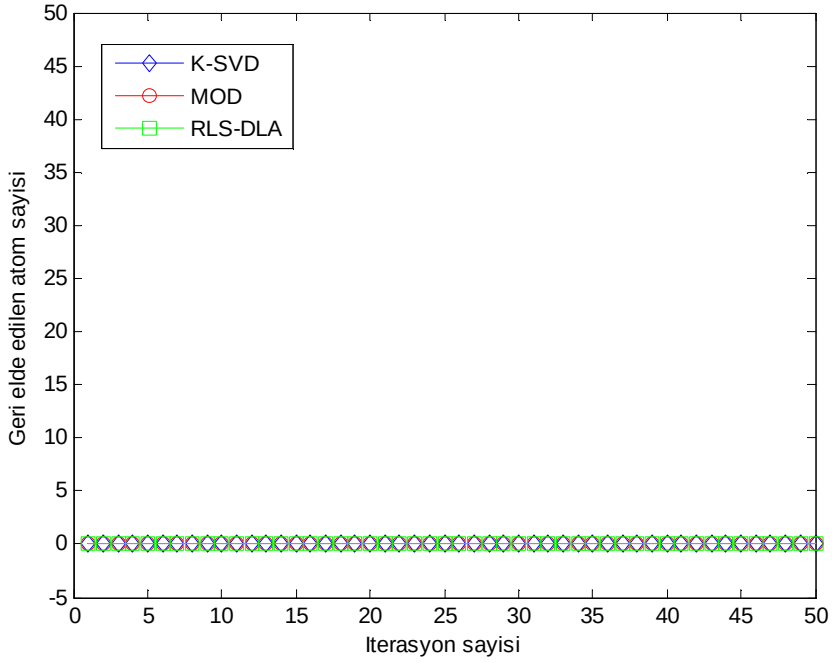


Şekil 5.11 : Veri matrisindeki örnek sayısı 1500, seyreklik değeri 5, işaret gürültü oranı 20 dB, sözlük matrisinin sütun sayısı 50, iterasyon sayısı 50 ve veri uzunluğu 20 şartlarında algoritmaların başarımlarının karşılaştırılması.



Şekil 5.12 : Veri matrisindeki örnek sayısı 1500, seyreklik değeri 6, işaret gürültü oranı 20 dB, sözlük matrisinin sütun sayısı 50, iterasyon sayısı 50 ve veri uzunluğu 20 şartlarında algoritmaların başarımlarının karşılaştırılması.

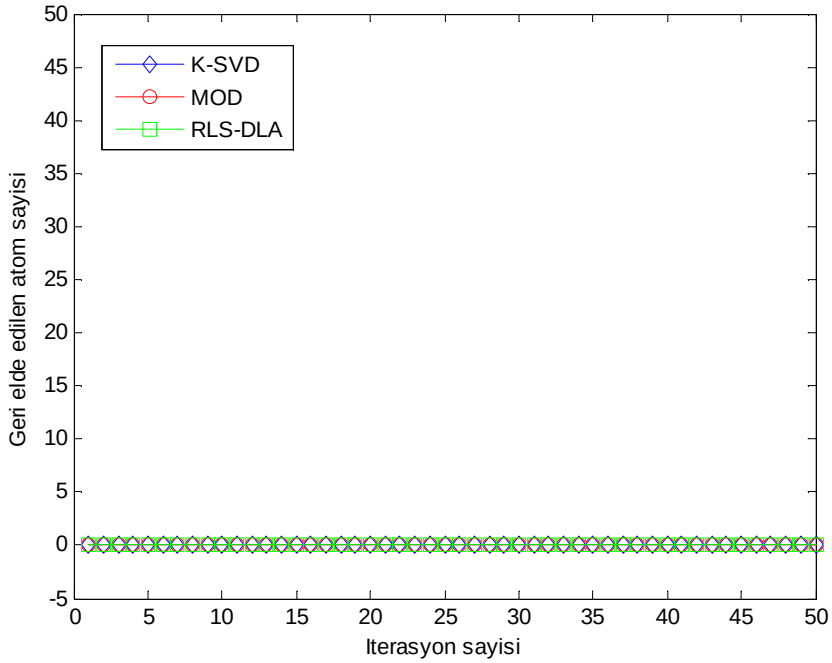
7 ve üzerindeki seyreklik değerlerinde geri elde edilen atom sayısı sıfır olduğu için bundan sonraki grafikleri sadece s=7 ve 20 için çizdirilmiştir. Şekil 5.13 ve Şekil 5.14 yardımıyla 20 veri uzunluğuna sahip bir veri kümesi için 7 ve üzerindeki seyrekliğin mümkün olmayacağı görülecektir.



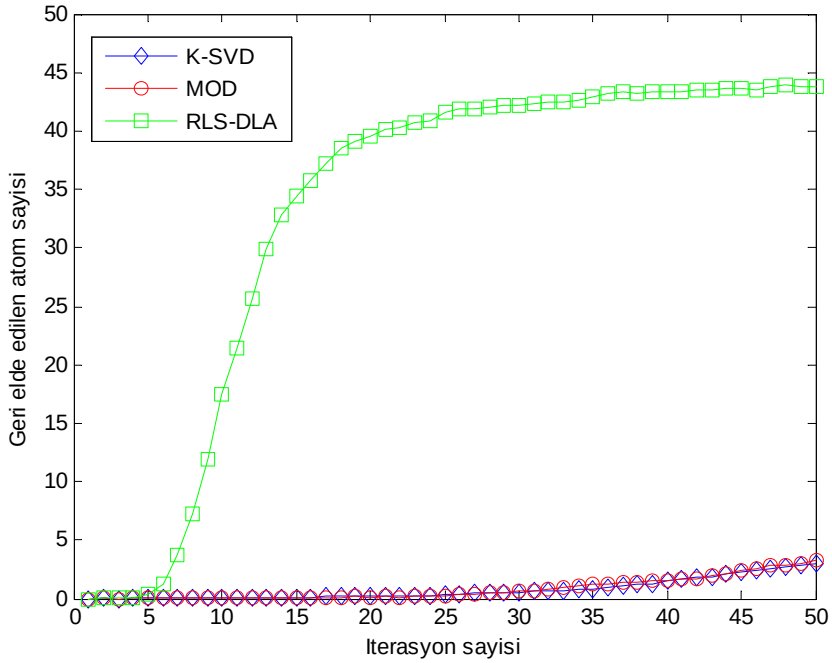
Şekil 5.13 : Veri matrisindeki örnek sayısı 1500, seyreklik değeri 7, işaret gürültü oranı 20 dB, sözlük matrisinin sütun sayısı 50, iterasyon sayısı 50 ve veri uzunluğu 20 şartlarında algoritmaların başarımlarının karşılaştırılması.

Şekillerden de görüldüğü üzere, 7 ve üzerindeki hiçbir seyreklik değerinde orijinal sözlüğe uygun bir sözlük elde edilememiştir. Ayrıca şekillerden de görülebileceği gibi tüm seyreklik değerlerinde RLS-DLA algoritması, diğer yöntemlere göre daha başarılı sonuçlar elde edilmesini sağlamıştır. Örneğin seyreklik değerinin 5 ve 6 olduğu durumda, diğer algoritmalarda hemen hemen hiçbir atom geri elde edilememişken RLS-DLA algoritması sonucunda belirli sayıda atom geri elde edilebilmektedir. Seyreklik değeri 5 olan durum için RLS-DLA algoritmasının 50 iterasyon sonucunda atomların ~%90 kadarı geri elde edilebilmiştir.

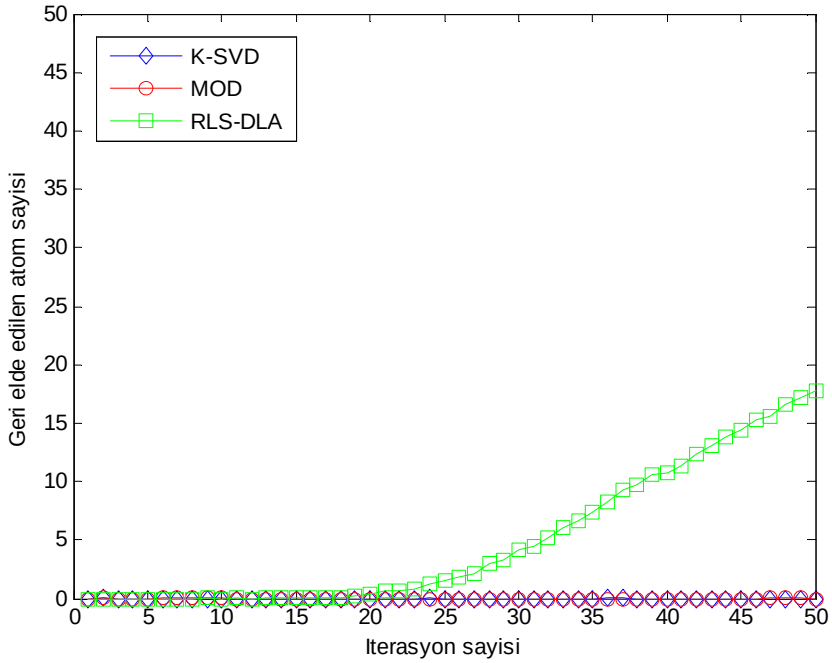
Bunun yanında durum sadece veri kümesindeki örnek sayısı 1500 iken değil daha küçük veri setlerinde de aynıdır. Şekil 5.15, Şekil 5.16 ve Şekil 5.17; N=500 değerindeyken çizdirilmiş ve burada da seyreklik değeri 8 gibi bir değerde sözlükteki hiçbir atomun geri elde edilemediği görülmüştür.



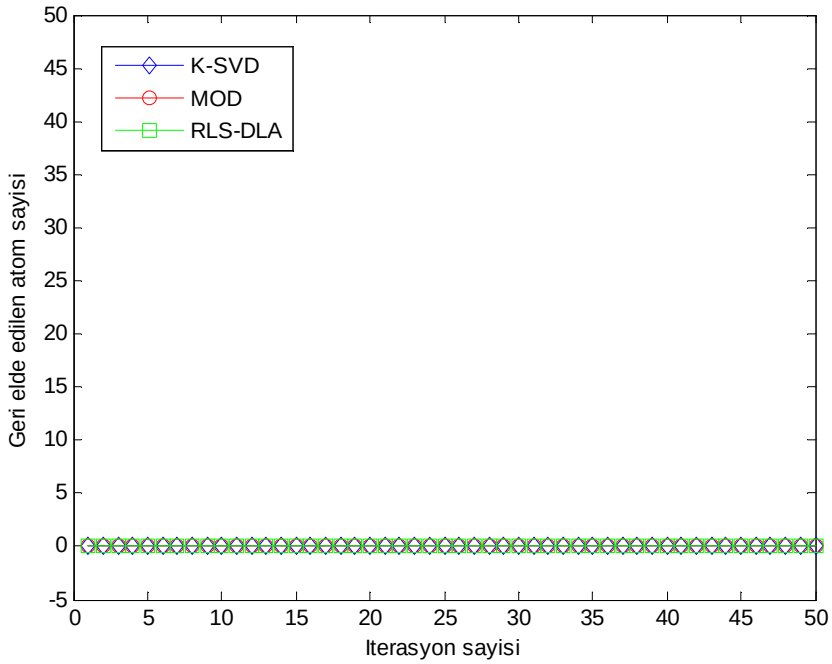
Şekil 5.14 : Veri matrisindeki örnek sayısı 1500, seyreklik değeri 20, işaret gürültü oranı 20 dB, sözlük matrisinin sütun sayısı 50, iterasyon sayısı 50 ve veri uzunluğu 20 şartlarında algoritmaların başarımlarının karşılaştırılması.



Şekil 5.15 : Veri matrisindeki örnek sayısı 500, seyreklik değeri 4, işaret gürültü oranı 20 dB, sözlük matrisinin sütun sayısı 50, iterasyon sayısı 50 ve veri uzunluğu 20 şartlarında algoritmaların başarımlarının karşılaştırılması.

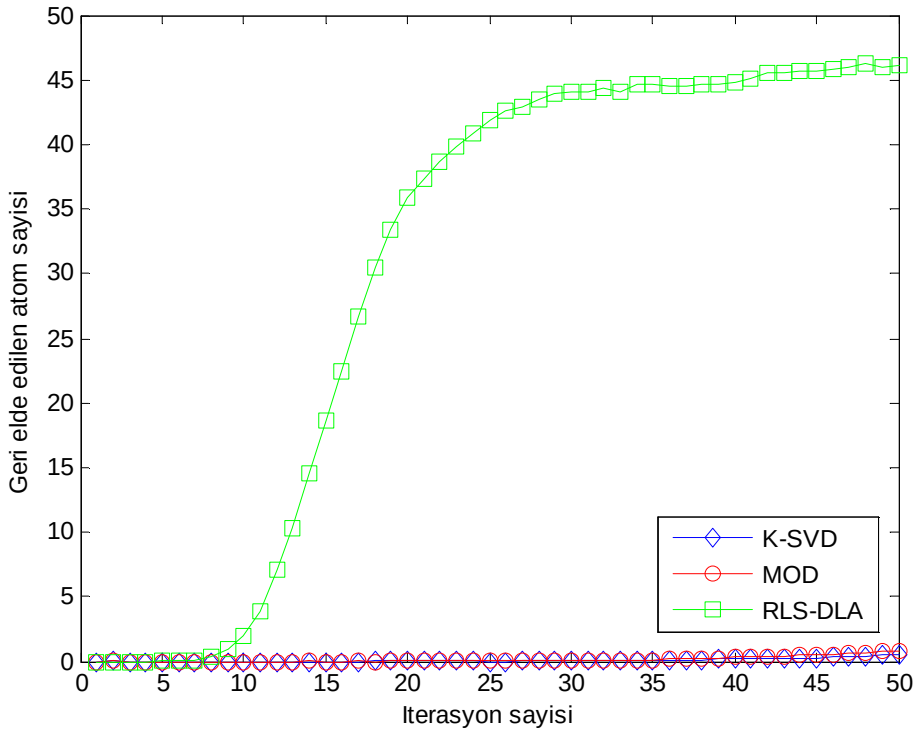


Şekil 5.16 : Veri matrisindeki örnek sayısı 500, seyreklik değeri 5, işaret gürültü oranı 20 dB, sözlük matrisinin sütun sayısı 50, iterasyon sayısı 50 ve veri uzunluğu 20 şartlarında algoritmaların başarımlarının karşılaştırılması.



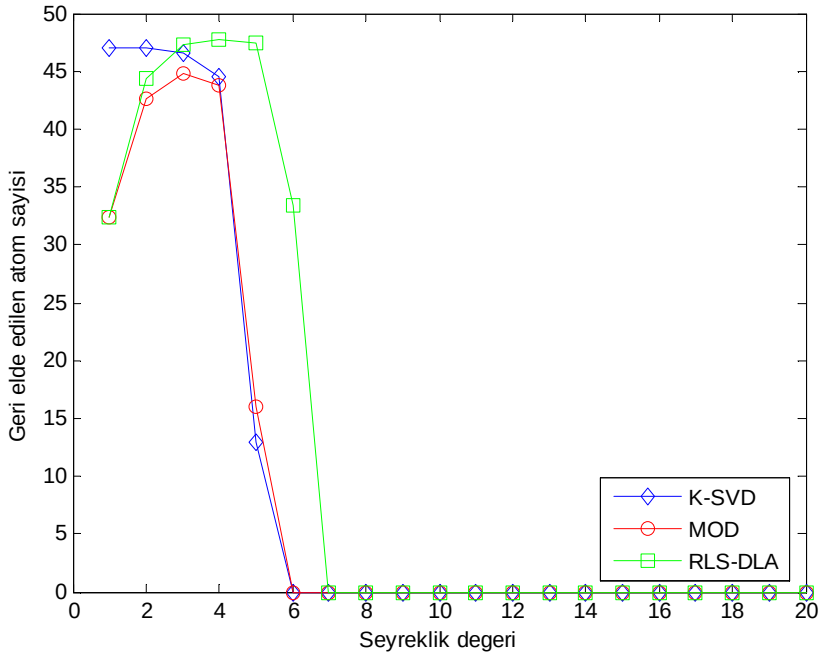
Şekil 5.17 : Veri matrisindeki örnek sayısı 500, seyreklik değeri 8, işaret gürültü oranı 20 dB, sözlük matrisinin sütun sayısı 50, iterasyon sayısı 50 ve veri uzunluğu 20 şartlarında algoritmaların başarımlarının karşılaştırılması.

Benzer durum veri kümesinin boyutu N=1000 olduğu durum içinde geçerlidir. Seyreklik değeri $s > 2$ olduğu müddetçe tüm veri uzunlukları için RLS-DLA algoritmasının performansı daha yüksektir. Örnek olarak Şekil 5.18'de seyreklik değeri 5 için sonuçlar çizdirilmiştir. Elde edilen grafikten de görüleceği üzere RLS-DLA algoritması K-SVD ve MOD algoritmalarına göre daha başarılı sonuçlar çıkmasını sağlamaktadır. MOD ve K-SVD algoritmalarında hiçbir atom geri elde edilemezken; RLS-DLA yöntemi ile 50 iterasyon sonucunda; sözlük atomlarının yaklaşık yarısı geri elde edilmiştir.



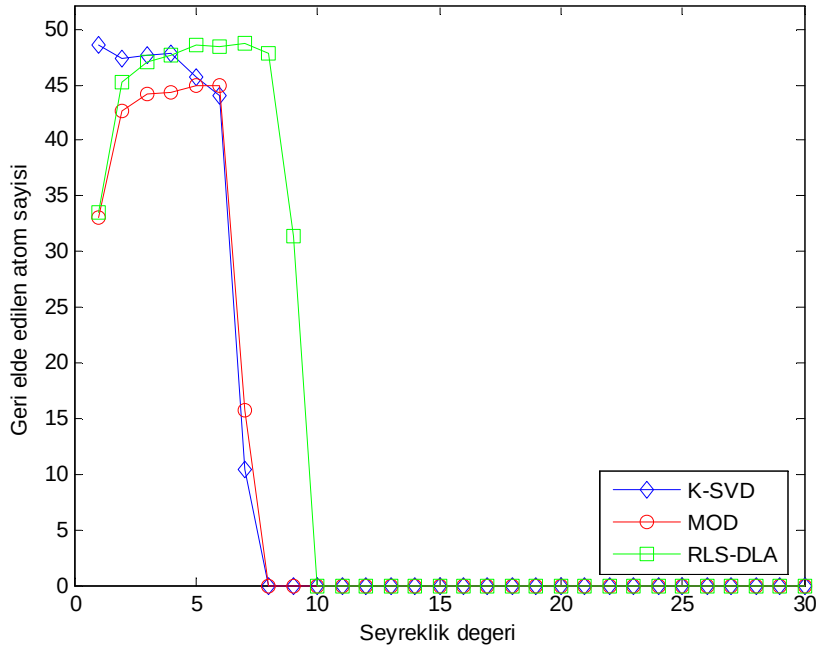
Şekil 5.18 : Veri matrisindeki örnek sayısı 1000, seyreklik değeri 5, işaret gürültü oranı 20 dB, sözlük matrisinin sütun sayısı 50, iterasyon sayısı 50 ve veri uzunluğu 20 şartlarında algoritmaların başarımlarının karşılaştırılması.

Detaylı şekilde çizdirilen bu grafiklerden sonra, s değerinin değişimine göre 50 iterasyon sonucunda geri elde edilen atom sayılarına ait grafik Şekil 5.19'da verilmiştir. Grafik tüm algoritmaların bir arada performanslarının çizdirilmesi ile oluşturulmuştur. Şekil 5.19'da görülebileceği gibi; tüm algoritmaların performanslarının daha yüksek olduğu bir seyreklik değeri mevcuttur.



Şekil 5.19 : Veri uzunluğu 20, seyreklik değeri değişimine göre algoritmaların başarımlarının karşılaştırılması.

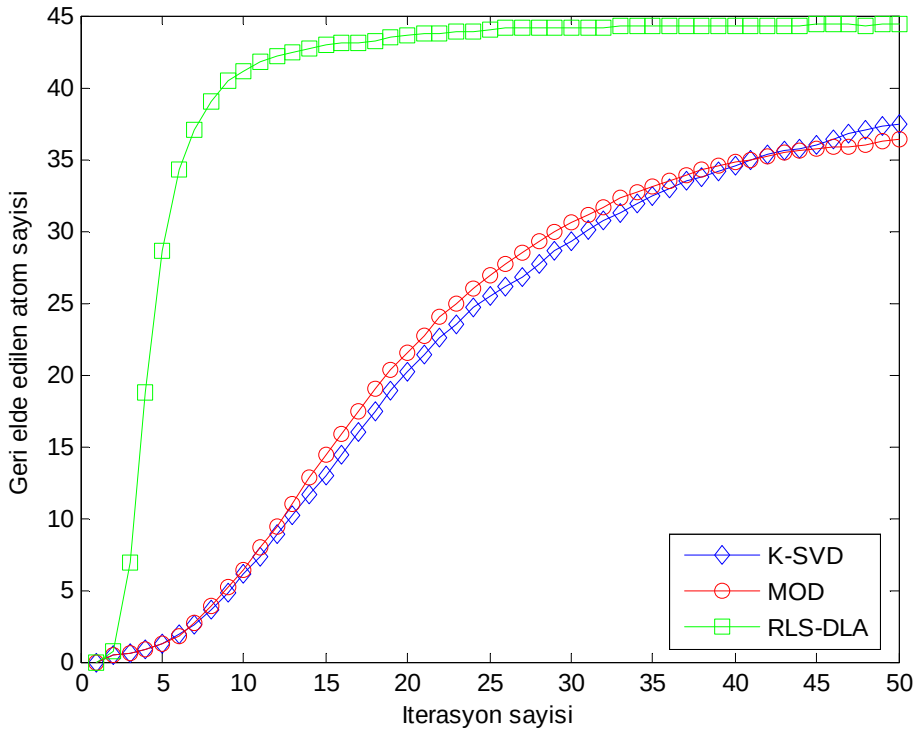
Benzer şekilde veri uzunluğu 30 iken de belirli bir seyreklik değerinden sonra sözlük matrisinin hiçbir atomunun geri elde edilemeyeceği görülecektir. Şekil 5.20 bu durumu göstermektedir.



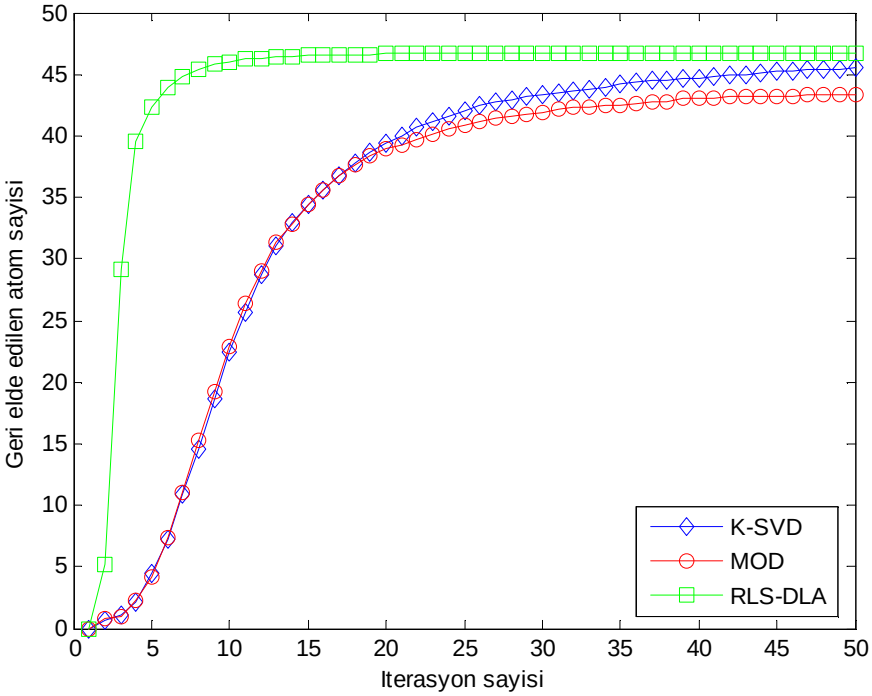
Şekil 5.20 : Veri kümesinin uzunluğu 30, seyreklik değeri değişimine göre algoritmaların başarımlarının karşılaştırılması.

Grafikler incelendiğinde KSVD algoritması dışındaki algoritmalar için yani MOD ve RLS-DLA için seyreklik değerinin bu veri kümesi için alabileceği optimum bir seyreklik değeri olduğu söylenebilir. 20 uzunluktaki bir veri kümesi için bu seyreklik değeri 3 şeklinde olmuştur. Ancak veri uzunluğunun, veri kümesindeki örnek sayısı ya da sözlük boyutu gibi faktörlerin değişmesi ile ideal seyreklik değeri değişiklik gösterecektir. Bu değişim Şekil 5.20'de veri uzunluğu 30 olarak değiştirilerek gözlenmiştir.

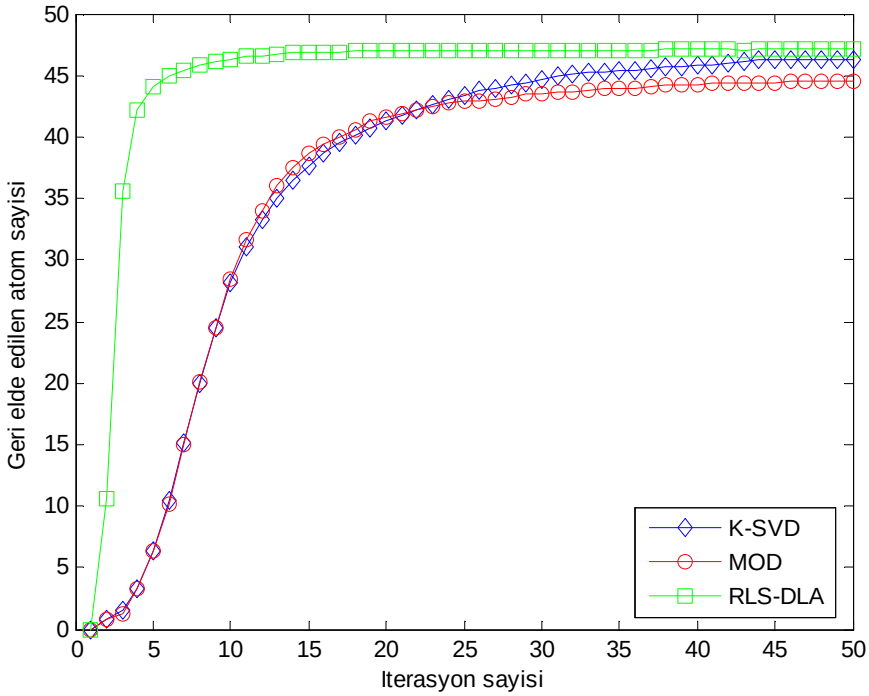
Çalışmanın bu bölümünde, veri matrisinin sütun boyutu N:500, N:1000 ve N:1500 için algoritmalar gerçeklenmiş ve Şekil 5.21, Şekil 5.22 ve Şekil 5.23 çizdirilmiştir. Bu deney sırasında diğer değişkenler; SNR değeri 20 dB, veri uzunluğu 20, sözlük sütun boyutu 50, seyreklik değeri 3 ve iterasyon sayısı 50 şeklindedir. İşlemler 100 defa tekrarlanarak ortalaması alınmıştır. Özellikle düşük N değerleri için RLS-DLA algoritmasının çok daha başarılı sonuçlar verdiği gözlenmiştir.



Şekil 5.21 : Veri matrisinin sütun sayısı 500, SNR değeri 20 dB, veri uzunluğu 20, sözlük sütun boyutu 50, seyreklik değeri 3 ve iterasyon sayısı 50 şartlarında algoritmaların başarımları.

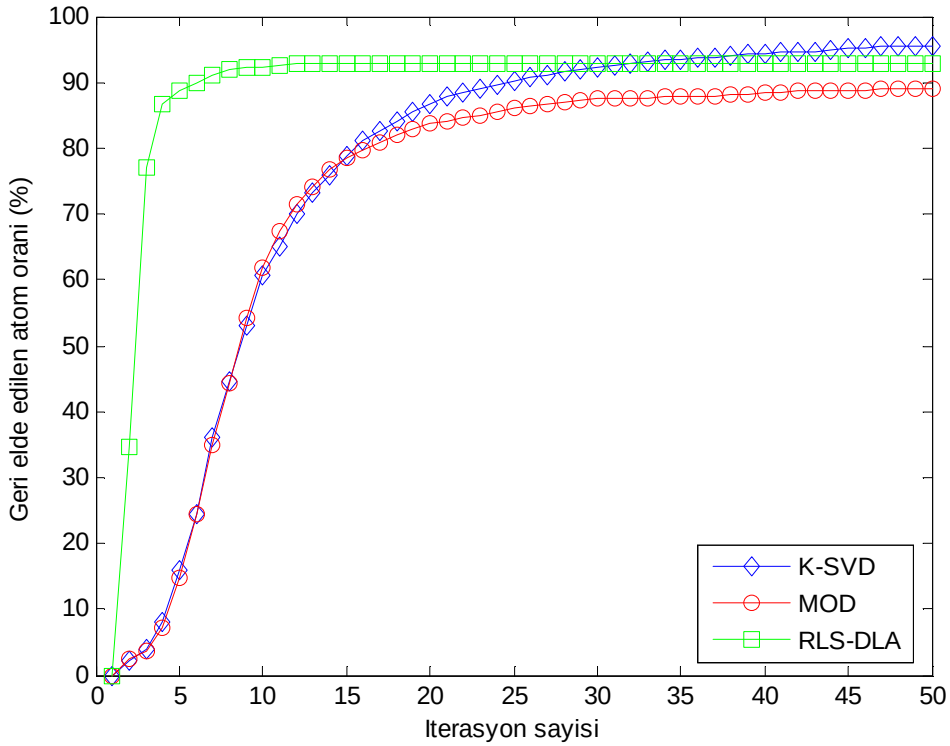


Şekil 5.22 : Veri matrisinin sütun sayısı 1000, SNR değeri 20 dB, veri uzunluğu 20, sözlük sütun boyutu 50, seyreklik değeri 3 ve iterasyon sayısı 50 şartlarında algoritmaların başarımları.

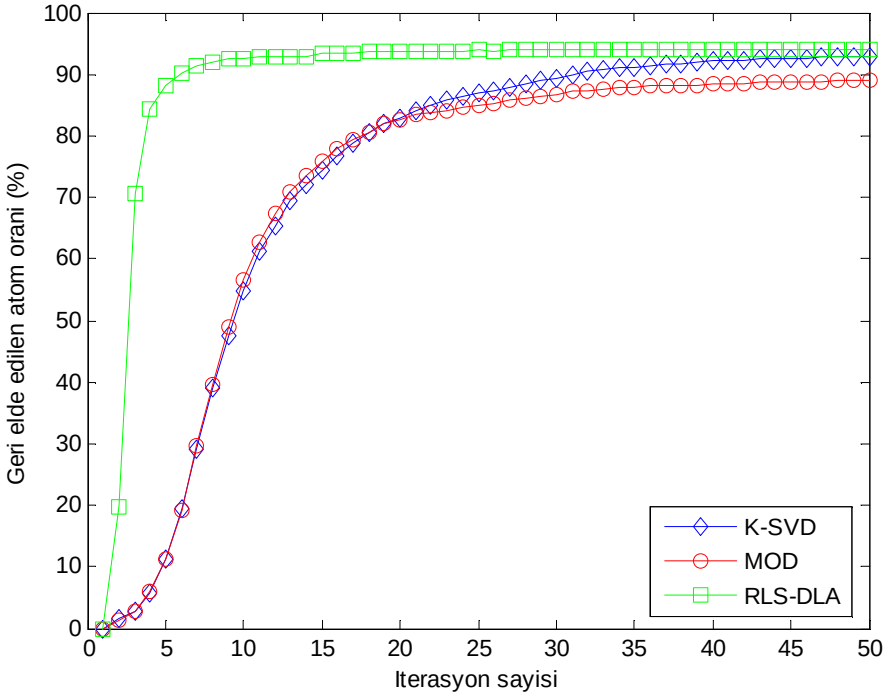


Şekil 5.23 : Veri matrisinin sütun sayısı 1500, SNR değeri 20 dB, veri uzunluğu 20, sözlük sütun boyutu 50, seyreklik değeri 3 ve iterasyon sayısı 50 şartlarında algoritmaların başarımları.

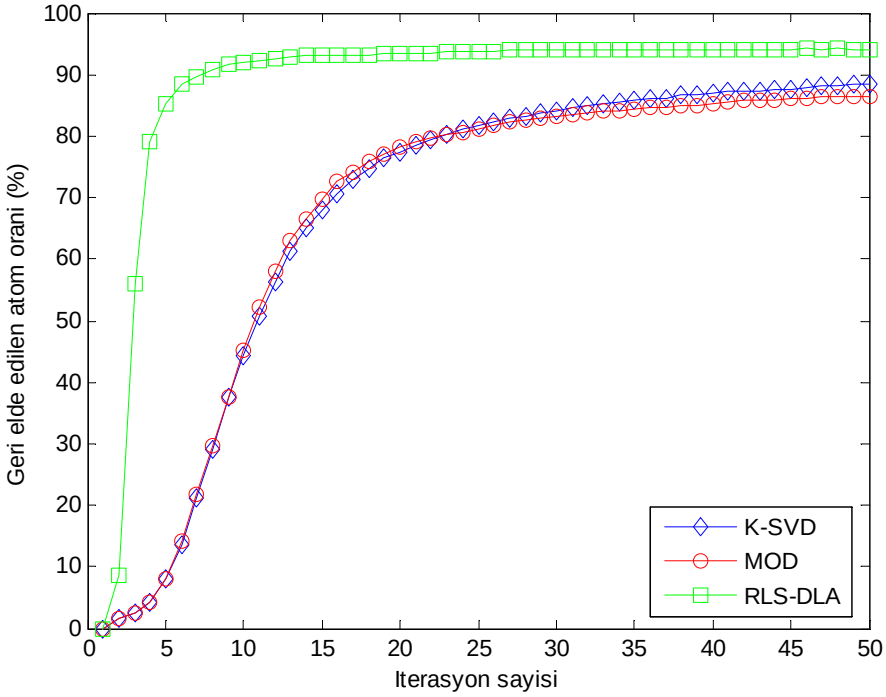
Çalışmanın bundan sonraki bölümünde seyrek gösterilim probleminde geriye kalan iki değişken olarak K sözlük sütun boyutunun ve veri uzunluğunun başarım sonuçlarına etkisi gözlenecektir. Öncelikle diğer değişkenlerin sabit olduğu durumda (N:1500, SNR: 20 dB, M:20, iterasyon sayısı: 50, s:3) K sözlük sütun boyutu 30, 50, 70, 90 olarak değiştirilmiştir. Elde edilen sonuçlar Şekil 5.24, Şekil 5.25, Şekil 5.26 ve Şekil 5.27'de yüzde oran cinsinden verilmiştir. Değişik sözlük sütun boyutları için de RLS-DLA algoritmasının diğer algoritmalara göre daha başarılı sonuçlar verdiği gözlenmiştir.



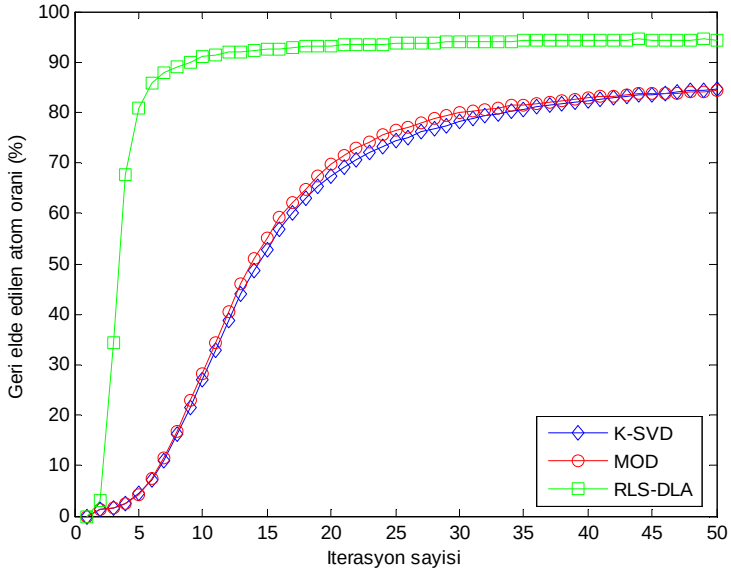
Şekil 5.24 : Sözlük matrisinin sütun sayısı (atom sayısı) 30 (N:1500, SNR: 20 dB, M:20, iterasyon sayısı: 50, s:3) iken algoritmaların başarımları.



Şekil 5.25 : Sözlük matrisinin sütun sayısı (atom sayısı) 50 (N:1500, SNR: 20 dB, M:20, iterasyon sayısı: 50, s:3) iken algoritmaların başarımları.

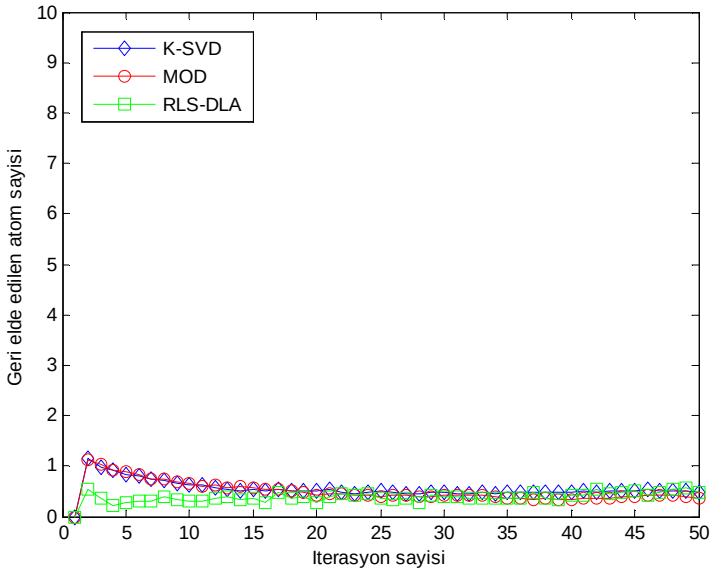


Şekil 5.26: Sözlük matrisinin sütun sayısı (atom sayısı) 70 (N:1500, SNR: 20 dB, M:20, iterasyon sayısı: 50, s:3) iken algoritmaların başarımları.



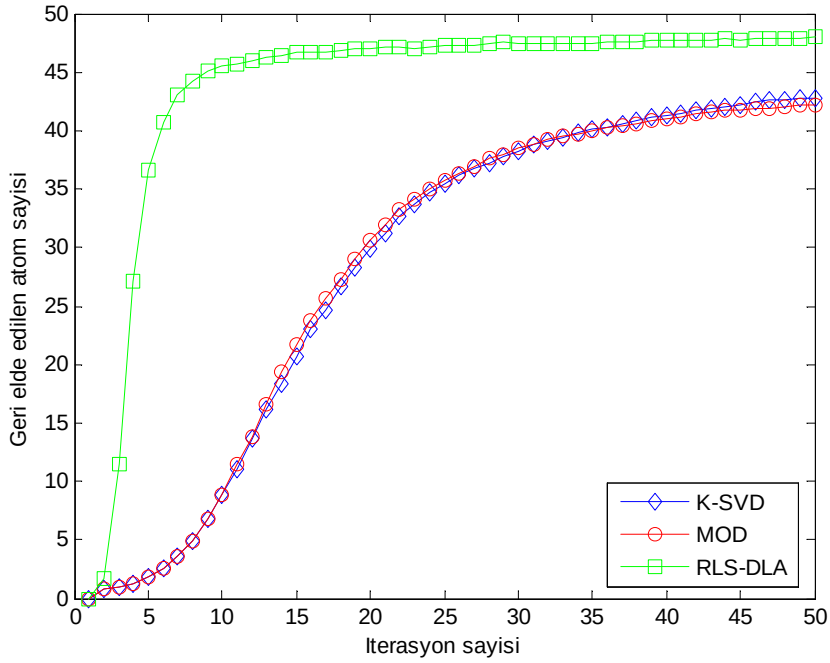
Şekil 5.27: Sözlük matrisinin sütun sayısı (atom sayısı) 90 (N:1500, SNR: 20 dB, M:20, iterasyon sayısı: 50, s:3) iken algoritmaların başarımları.

Bir diğer gözlemde *M* veri uzunluğu 7, 15, 30, 60 olarak değiştirilerek algoritmalar çalıştırılmıştır. Öncelikle *M*=7 için aşağıdaki Şekil 5.32 çizdirilmiştir. Ancak bu değer içen çıkan sonuçlar oldukça kötüdür. Bunun nedeni seyrekli değeri 3 sabit bırakılarak veri kümesinin boyutunun küçültülmesidir. Diğer değişkenler sabit kaldığı müddetçe küçük *M* değerleri için çıkan sonuçlar tüm algoritmalarda tatmin edici değildir.

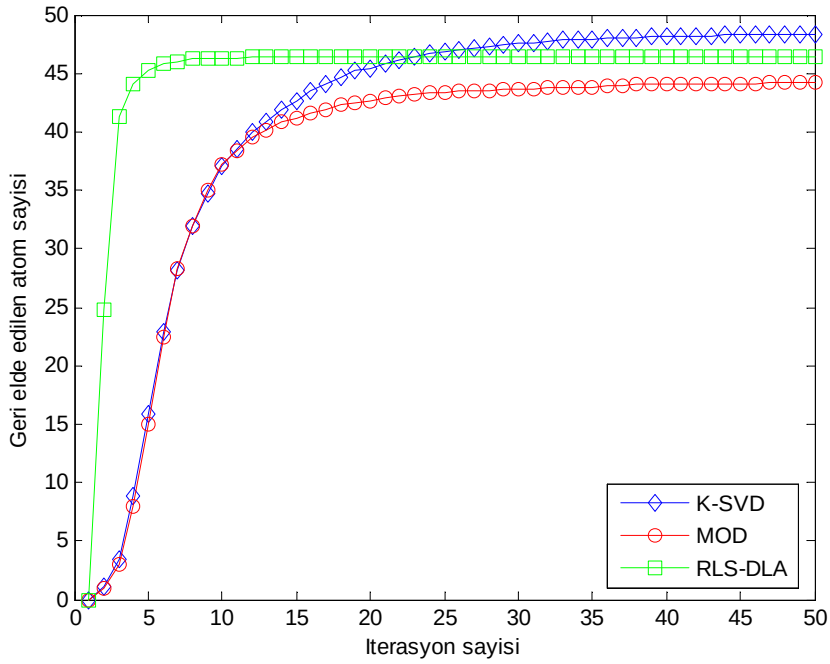


Şekil 5.28 : Veri uzunluğu 7 (N:1500, SNR: 20 dB, K: 50, iterasyon sayısı: 50, s:3) iken algoritmaların başarımları.

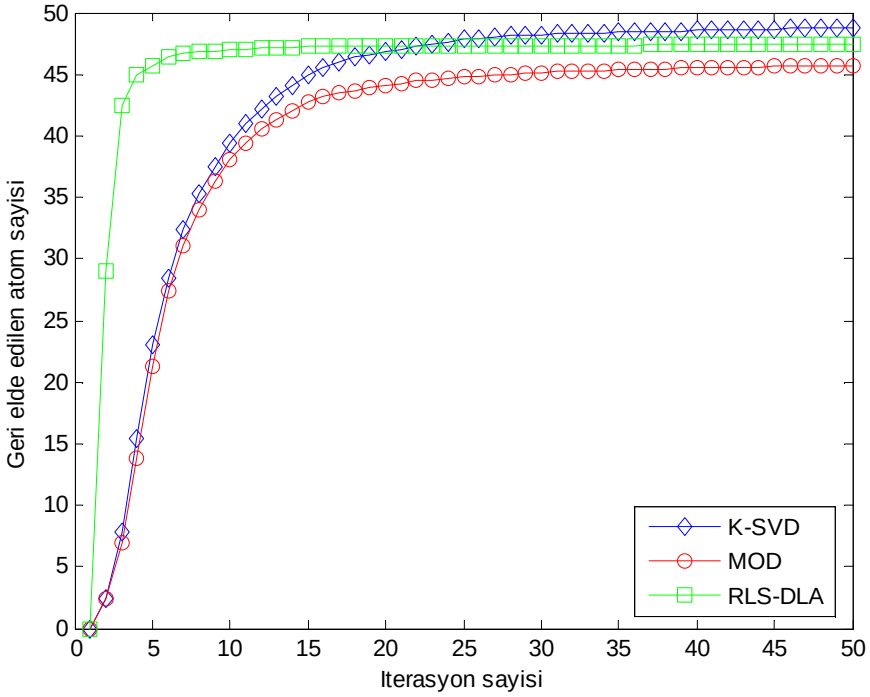
Benzer şekilde diğer veri uzunlukları içinde (M: 15, 30 ve 60 için) sonuçlar Şekil 5.29, Şekil 5.30 ve Şekil 5.31'de ifade edilmiştir. Seyreklik değeri ile veri kümesinin uzunluğu arasındaki ilişki bu gözlemlerde de ortaya çıkmıştır.



Şekil 5.29 : Veri uzunluğu 15 (N:1500, SNR: 20 dB, K: 50, iterasyon sayısı: 50, s:3) iken algoritmaların başarımları.



Şekil 5.30 : Veri uzunluğu 30 (N:1500, SNR: 20 dB, K: 50, iterasyon sayısı: 50, s:3) iken algoritmaların başarımları.



Şekil 5.31 : Veri uzunluğu 60 (N:1500, SNR: 20 dB, K: 50, iterasyon sayısı: 50, s:3) iken algoritmaların başarımları.

6. SONUÇLAR

Yüksek lisans tez çalışmasında, sözlük tanılama yöntemleri kullanılarak işaretlerin seyrek gösterilimi konusu üzerinde çalışılmıştır. Bu amaçla, sözlük tanılama yöntemleri ve işaretlerin seyrek gösteriliminde kullanılan vektör seçim algoritmaları incelenmiştir. İşaretlerin seyrek gösteriliminde kullanılabilecek sözlüklerin verinin kendisinden elde edilmesine imkan veren MOD ve K-SVD gibi sözlük öğrenme algoritmaları ile birlikte, sözlük tasarımında daha hızlı yakınsama sağlayan RLS-DLA yöntemi de MATLAB ortamında gerçeklenmiştir. Gerçeklemeler ile birlikte ortaya çıkan sonuçlar algoritmaların başarımlarının tespiti için kullanılmıştır. Bu sonuçlara göre RLS-DLA algoritması tüm veri uzunluklarında diğer yöntemlere göre çok daha az iterasyon sonucunda sözlük matrisine ulaşılmasını sağlamaktadır.

Bunun yanında işaretlerin seyrek gösterimlerini sağlarken kullandığımız değişkenlerin tümü algoritmaların başarımları üzerine etkisi bulunmaktadır. Bunlardan özellikle seyreklik değerinin mevcut veri uzunluğunu bağlı olarak başarımı etkilediği söylenebilir. Gerçeklenen 20 veri uzunluğuna sahip bir örnek için 7 seyreklik değerinden sonra işaretin seyrek gösterimini bulmak mümkün olmamaktadır. Bunun yanında veri uzunluğu arttıkça erişilebilecek seyreklik değerinde artış olacağı açıktır.

Sözlük matrisinin sütun sayısı ve veri uzunluğu gibi faktörlerin hepsi algoritmaların başarımlarını bir şekilde etkilemektedir. Ancak düşük seyreklik değerleri dışındaki tüm durumlarda RLS-DLA algoritması diğer algoritmalardan daha başarılı sonuçlar elde edilmesini sağlamaktadır. Bu başarım sözlük matrisi için elde edilen atom sayısından kolaylıkla görülmektedir.

Yüksek lisans tez çalışmasında gerçeklenen bu yöntemler ile birlikte işaretlerin seyrek gösterilimlerine karşı oluşan ilginin giderek artacağı ve yeni sözlük algoritmalarının bulunmasına yönelik çalışmaların devam edeceği söylenebilir.

KAYNAKLAR

[1] **K. Skretting ve K. Engan**, "Recursive least squares dictionary learning algorithm," *Signal Processing, IEEE Transactions on*, vol 58, no. 4, Nisan 2010, sayfa 2121-2130.

[2] **M. Aharon, M. Elad ve A. Bruckstein**, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *Signal Processing, IEEE Transactions on*, vol. 54, no. 11, Kasım 2006, sayfa 4311-4322.

[3] **M. Aharon**, "Overcomplete Dictionaries for Sparse Represantation of Signals," Phd. Thesis, *Israel Institute of Technology, Haifa,* Kasım 2006.

[4] **S. Mallat ve Z. Zhang**, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. Signal Processing*, 41(12), 1993, sayfa 3397-3415.

[5] **S. Chen, S.A. Billings ve W. Luo**, "Orthogonal least squares methods and their application to non-linear system identification", *International Journal of Control*, 50(5), 1989, sayfa 1873-96.

[6] **S. S. Chen, D. L. Donoho, ve M. A. Saunders**, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comput.*, vol. 20, no.1, 1998, sayfa 33-61.

[7] **I.F. Gorodnitsky ve B.D. Rao**, "Sparse signal reconstruction from limited data using FOCUSS: a re-weighted minimum norm algorithm," *Signal Processing, IEEE Transactions on* , vol.45, no.3, Mart 1997, sayfa 600-616.

[8] **R. Rubinstein, A. M. Bruckstein, ve M. Elad**, "Dictionaries for sparse representation modeling," *Signal Processing, IEEE Transactions on*, vol. 98, no. 6, 2010, sayfa 1045-1057.

[9] **M.S. Lewicki ve T.J. Sejnowski**, "Learning overcomplete representations," *Neural Comp.*, vol. 12, 2000, sayfa 337-365.

[10] **B.A. Olshausen ve D.J. Field**, "Natural image statistics and effcient coding," *Network: Computation in Neural Systems*, 7(2), 1996, sayfa 333-9.

[11] **M. Elad**, "*Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*," Springer New York Dordrecht Heidelberg London, 2010.

[12] **K. Engan, S. O. Aase, ve J. H. Husøy**, "Multi-frame compression: theory and design," *Signal Process.*, vol. 80, no. 10, 2000, sayfa 2121-2140.

[13] **K. Engan, S.O. Aase, ve J. H. Husøy**, "Method of optimal directions for frame design," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, 1999, sayfa 2443-2446.

[14] **K. Kreutz-Delgado, B. D. Rao, ve K. Engan**, "Novel algorithms for learning overcomplete dictionaries," *Proc. Conf. Signals, Systems, ve Computer Record of the Thirty-Third Asilomar Conf.*, vol. 2, 1999, sayfa 971-975.

[15] **A. Joel, ve J. W. Stephen**, "Computational Methods for Sparse Solution of Linear Inverse Problems," vol. 98, no. 6, Haziran 2010, sayfa 948-958.

[16] **E. J. Candès, M. B. Wakin, ve S. P. Boyd**, "Enhancing sparsity by reweighted l-1 minimization," *Journal of Fourier Analysis and Applications*, vol. 14, 2008, sayfa 877-905.

[17] **K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan**, **T. W. Lee ve T. J. Sejnowski**, "Dictionary learning algorithms for sparse representation," *Neural Comput.*, vol. 15, no. 2, 2003, sayfa 349-396.

[18] **M. Yaghoobi, T. Blumensath, ve M. E. Davies**, "Dictionary learning for sparse approximations with the majorization method," vol. 57, no. 6, 2009, sayfa 2178-2191.

[19] **K. Engan, K. Skretting, ve J. H. Husøy**, "Family of iterative LS-based dictionary learning algorithms, ILS-DLA, for sparse signal representation," *Digit. Signal Process.*, vol. 17, no.1, 2007, sayfa 32-49.

[20] **R. Rubinstein, M. Zibulevsky, ve M. Elad**, "Double sparsity: Learning sparse dictionaries for sparse signal approximation," vol. 58, no. 3, 2010, sayfa 1553-1564.

[21] **S. Haykin**, "*Adaptive Filter Theory (4th Ed.),*" Prentice Hall, 2001.

ÖZGEÇMİŞ

EMRAH YAVUZ

18 Ağustos 1985 de Bakırköy/İstanbul da doğdu. İlköğrenimini Çatalca Ferhatpaşa İlköğretim Okulu'nda; orta öğrenimini Pertevniyal Anadolu Lisesi'nde tamamladı. Lisans öğreniminde İstanbul Teknik Üniversitesi Elektrik-Elektronik Fakültesi Telekomünikasyon Mühendisliği bölümünü bitirdi. Yüksek lisans eğitimine aynı bölümde devam etti. 2008 yılından itibaren ARÇELİK A.Ş. Elektronik İşletmesinde yazılım mühendisi görevinde çalışmaktadır. İngilizce, iyi derecede; Almanca başlangıç düzeyinde bilmektedir.

Telekomünikasyon alanında sinyal işleme ve mobil haberleşme ilgi alanları arasında yer alır. Tarih, ekonomi ve sinema başlıca hobileridir.