T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ
ANABİLİM DALI

ÇEVRİMİÇİ DESTEK VEKTÖR MAKİNELERİ TABANLI MODEL ÖNGÖRÜLÜ DENETİM

YÜKSEK LİSANS TEZİ

MERVE TOPALOĞLU

DENİZLİ, AĞUSTOS - 2014

T.C.
PAMUKKALE ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ
ANABİLİM DALI

ÇEVRİMİÇİ DESTEK VEKTÖR MAKİNELERİ TABANLI MODEL ÖNGÖRÜLÜ DENETİM

YÜKSEK LİSANS TEZİ

MERVE TOPALOĞLU

DENİZLİ, AĞUSTOS - 2014

KABUL VE ONAY SAYFASI

Merve TOPALOĞLU tarafından hazırlanan "**Çevrimiçi Destek Vektör Makineleri Tabanlı Model Öngörülü Denetim**" adlı tez çalışmasının savunma sınavı 29.08.2014 tarihinde yapılmış olup aşağıda verilen jüri tarafından oy birliği / oy çokluğu ile Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Elektrik-Elektronik Mühendisliği Anabilim Dalı Yüksek Lisans Tezi olarak kabul edilmiştir.

Jüri Üyeleri	İmza
Danışman Prof. Dr. Serdar İPLİKÇİ	.....
Üye Doç. Dr. Kadir KAVAKLIOĞLU	.....
Üye Yrd. Doç. Dr. Selami BEYHAN	.....

Pamukkale Üniversitesi Fen Bilimleri Enstitüsü Yönetim Kurulu'nun 10/09/2014. tarih ve ...36/24... sayılı kararıyla onaylanmıştır..

.....

Prof. Dr. Orhan KARABULUT

Fen Bilimleri Enstitüsü Müdürü

Bu tezin tasarımı, hazırlanması, yürütülmesi, araştırmalarının yapılması ve bulgularının analizlerinde bilimsel etiğe ve akademik kurallara özenle riayet edildiğini; bu çalışmanın doğrudan birincil ürünü olmayan bulguların, verilerin ve materyallerin bilimsel etiğe uygun olarak kaynak gösterildiğini ve alıntı yapılan çalışmalara atfedildiğine beyan ederim.

Merve TOPALOĞLU

ÖZET

ÇEVRİMİÇİ DESTEK VEKTÖR MAKİNELERİ TABANLI MODEL ÖNGÖRÜLÜ DENETİM

YÜKSEK LİSANS TEZİ

MERVE TOPALOĞLU

PAMUKKALE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ELEKTRİK-ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI

(TEZ DANIŞMANI: PROF.DR. SERDAR İPLİKÇİ)

DENİZLİ, AĞUSTOS - 2014

Tez çalışmasında, destek vektör makinelerinin (SVM) çevrimiçi eğitimi için önerilmiş bir yöntem olan çevrimiçi destek vektör bağlanımı algoritmasıyla, destek vektörü yapılarına dayalı bir denetim yöntemi olan daha önceden önerilmiş Destek Vektör Makineleri Tabanlı Model Öngörülü Denetim (SVM-Tabanlı MPC) yöntemini birleştiren çevrimiçi SVM-Tabanlı MPC yöntemi hem doğrusal sistemlere hem de doğrusal olmayan sistemlere uygulanmıştır. Bu yöntem, denetimi yapılacak sistemin boş bir modeliyle başlayarak modelleme ve denetim süreçlerini paralel olarak yapmaktadır. Benzetim sonuçları, çevrimiçi SVM-Tabanlı MPC yönteminin, hem doğrusal sistemlerin hem de doğrusal olmayan sistemlerin denetiminde oldukça iyi bir başarımı olduğunu göstermiştir.

ANAHTAR KELİMELER: model öngörülü denetim; destek vektör makineleri; modelleme ve tahmin

ABSTRACT

ONLINE SUPPORT VECTOR MACHINES BASED MODEL PREDICTIVE CONTROL
MSC THESIS
MERVE TOPALOĞLU
PAMUKKALE UNIVERSITY INSTITUTE OF SCIENCE
ELECTRİCAL AND ELECTRONİCS ENGİNEERİNG

(SUPERVISOR: PROF.DR. SERDAR İPLİKÇİ)

DENİZLİ, AUGUST 2014

In this thesis, an online support vector machines (SVM) training method, referred to as online support vector regression algorithm, previously proposed support vector machines-based model predictive control (SVM-Based MPC) architecture, combines online SVM-based MPC method has been applied to both linear systems and nonlinear systems. Starting with an initially empty SVM model of the unknown plant, the proposed online SVM-based MPC method performs the modelling and control tasks simultaneously. The simulation results on both linear systems and nonlinear systems have revealed that the proposed method provides an excellent control quality.

KEYWORDS: model predictive control; support vector machines; modelling and prediction

İÇİNDEKİLER

Sayfa

ŞEKİL LİSTESİ

Sayfa

SEMBOL LİSTESİ

$\mathbf{s}$: Adım aralığı
$\mathbf{p}$: Arama yönü
$\boldsymbol{\nabla}$: Gradyan vektörü
$\boldsymbol{\nabla^2}$: Hessian matrisi
$\mathbf{I}$: Birim matris
$\mathbf{R}$: Üst üçgen matris
$\mathcal{N}(\mathbf{A})$: A matrisinin boş uzayı (Null Space)
$\mathcal{R}(\mathbf{A})$: A matrisinin Range uzayı
$\boldsymbol{\mathcal{L}}$: Lagrange fonksiyonu
$\mathbf{m}$: Geometrik marjin
$\mathbf{w}$: Ağırlık
$\mathbf{b}$: Eşik değeri
$\mathbf{C}$: Ceza parametresi
$\boldsymbol{\xi}$: Sınıflandırıcının yaptığı hata
$\boldsymbol{\alpha}$: Lagrange çarpanları
$\boldsymbol{\beta}$: Lagrange çarpanları
$\boldsymbol{\mathcal{F}}$: Öznitelik uzayı
$\mathbf{Q}$: Kernel fonksiyonu
$\boldsymbol{\phi}(.)$: Dönüşüm fonksiyonu
$\boldsymbol{\sigma}$: Genişlik parametresi
$\boldsymbol{\varepsilon}$: Hata toleransı
$\boldsymbol{\mu}$: Lagrange çarpanları
$\boldsymbol{\theta}$: Ağırlıklar
$\mathbf{L_{c1}}$: Yeni örneğin destek vektörü kümesine olan uzaklığı
$\mathbf{L_{c2}}$: Yeni örneğin hata kümesine olan uzaklığı
$\mathbf{L_S}$: Her bir destek vektörü kümesi örneğinin kalan kümesine ya da hata kümesine olan uzaklığı
$\mathbf{L_E}$: Hata kümesindeki her bir örneğin destek vektörü kümesine olan uzaklığı
$\mathbf{L_R}$: Kalan kümesindeki her bir örneğinin destek vektörü kümesine olan uzaklığı
$\mathbf{L_c}$: Yeni örneğin kalan kümesine olan uzaklığı
$\mathbf{u_n}$: n. zaman indeksi anında sisteme uygulanan denetim işareti
$\mathbf{y_n}$: n. zaman indeksi anında sistemin çıkışı
$\mathbf{n_u}$: Modelde yer alan geçmiş denetim işareti sayısı
$\mathbf{n_y}$: Modelde yer alan geçmiş çıkış işareti sayısı
$\hat{\mathbf{y}}$: Model çıkışı
$\tilde{\mathbf{y}}$: Sistem tarasından takip edilmesi istenen referans işareti
$\mathbf{J}$: Başarım göstergesi
$\mathbf{N_1}$: En kısa bedel ufku
$\mathbf{N_2}$: En uzun bedel ufku
$\mathbf{N_u}$: Denetim ufku
$\boldsymbol{\lambda}$: Ağırlık faktörü
$\boldsymbol{\delta_{i,j}}$: Kronecker Delta fonksiyonu
$\boldsymbol{\delta_1}(.)$: Birim basamak fonksiyonu
$\boldsymbol{\sigma_y}$: Ölçülen işaretin değişinti değeri

$\boldsymbol{\sigma_v}$	:	Eklenen gürültünün değişinti değeri
$\boldsymbol{A_{Cond}}$	:	Geri akma cihazında tutulan molar birikim
$\boldsymbol{A_{Tray}}$	:	Bölgelerdeki molar birikim
$\boldsymbol{A_{Reboiler}}$	:	Kazanda tutulan molar birikim
$\boldsymbol{F}$	:	Besleme molar akış hızı
$\boldsymbol{D}$	:	Damıtılmış ürün molar akış hızı
$\boldsymbol{B}$	:	Alt ürün molar akış hızı
$\boldsymbol{L_1}$	:	Zenginleştirme bölgesindeki sıvının molar akış hızı
$\boldsymbol{L_2}$	:	Sıyırma bölgesindeki sıvının molar akış hızı
$\boldsymbol{V}$	:	Buhar molar akış hızı
$\boldsymbol{RR}$	:	Geri akma oranı
$\boldsymbol{\propto_{A,B}}$	:	Uçuculuk
$\boldsymbol{x_{A,i}}$	:	A bileşiminin i. bölgedeki sıvı mol kesri
$\boldsymbol{y_{A,i}}$	:	A bileşiminin i. bölgedeki buhar mol kesri
$\boldsymbol{\sigma_y}$	:	Ölçülen işaretin değişinti değeri
$\boldsymbol{\sigma_v}$	:	Eklenen gürültünün değişinti değeri
$\boldsymbol{K_s}$	:	Yay katsayısı
$\boldsymbol{J_h}$	:	Merkez ataleti
$\boldsymbol{m}$	:	Yük ağırlığı
$\boldsymbol{g}$	:	Yerçekimi
$\boldsymbol{h}$	:	Yükseklik
$\boldsymbol{K_m}$	:	Motor sabiti
$\boldsymbol{K_g}$	:	Dişli oranı
$\boldsymbol{J_l}$	:	Yük ataleti
$\boldsymbol{R_m}$	:	Motor direnci

KISALTMALAR LİSTESİ

VMM : Değişken Metrik Yöntemler (Variable Metric Methods)
DFP : Davidson-Fletcher-Powell
BFGS : Broydon-Fletcher-Goldforb-Shanno
GN : Gauss Newton
LM : Levenberg Marquardt
QP : Karesel Programlama (Quadratic Programming)
SVM : Destek Vektör Makineleri (Support Vector Machines)
MMC : En Büyük Marjinli Sınıflandırıcı (Maximum Margin Classifier)
KÇ : Karar Çizgisi
KE : Karar Eğrisi
KKT : Karush-Kuhn-Tucker
GPC : Genelleştirilmiş Öngörülü Kontrol (Generalized Predictive Control)
MPC : Model Tabanlı Öngörülü Kontrol (Model Based Predictive Control)
NARX : Nonlinear Auto Regressive Exogenous
CFM : Bedel Fonksiyonu Minimizasyon (Cost Function Minimization)
RBF : Radyal Tabanlı Fonksiyon (Radial Based Function)
OSAPE : Bir Adım Sonrası Tahmin Hatası (One Step Ahead Prediction Error)

ÖNSÖZ

Çalışma boyunca bilimsel katkıları ile bana yardımcı olan, eğitimim süresince yardımlarını esirgemeyen, tez danışmanım ve hocam Prof. Dr. Serdar İPLİKÇİ'ye, bilgi ve deneyimlerinden yararlandığım Yard. Doç. Dr. Selami BEYHAN'a ve Doç. Dr. Kadir KAVAKLIOĞLU'na en içten teşekkür ve saygılarımı sunarım.

Hayatımın her alanında bana koşulsuz destek olan aileme tüm kalbimle teşekkür ederim.

1. GİRİŞ

Teorisi oldukça sağlam bir zemine oturan Genelleştirilmiş Öngörülü Denetim (Clarke ve diğ. 1987) (*Generalized Predictive Control-GPC*) yöntemi Model Tabanlı Öngörülü Denetim (*Model-Based Predictive Control-MPC*) teknikleri sınıfına dahildir. MPC teknikleri yaklaşık 30 yıldır açık-çevrim kararsız sistemlerin ve parametreleri veya ölü zamanları zamanla değişen (Clarke ve Mohtadi 1989) sistemlerin denetiminde dayanıklı bir yöntem olduklarını ispat ederek endüstriyel süreçlerin (Richalet 1993) denetiminin yanında kimyadan havacılığa kadar değişik alanlarda da kullanılmışlardır (Qin ve Badgwell 2003).

Literatürdeki ilk MPC tekniğinin (Richalet ve diğ. 1978) önerilmesinden sonra geliştirilen pek çok MPC yöntemlerinin içerisinde belkide en yaygın olanı GPC'dir (Clarke ve diğ. 1987). Yine de tüm MPC teknikleri aynı fikre dayanır: Denetimi yapılacak sistemin modelini kullanarak elde edilen ileri yönelik tahmine dayanarak, her bir örnekleme anında, sonlu-ufuklu açık-çevrimli bir en iyileme problemi çözülerek bir dizi denetim işareti elde edilir ve dizinin ilk elemanı sisteme uygulanır. Model tabanlı tekniklerde denetimi yapılacak sistemin modeli çok önemli rol oynadığından pek çok doğrusal olmayan modelleme yöntemleri uzunca zamandır kullanılmaktadır. Son zamanlarda işlemsel zeka alanındaki hızlı gelişmelere paralel olarak, GPC döngüsünde kullanılacak modelin elde edilmesinde yapay sinir ağları, bulanık sistemler, hibrit sistemler ve genetik algoritmalar (Martinez ve diğ. 1998) gibi esnek bilgi işlem araçlarından yararlanılmıştır.

Bağlanım problemini çözerek doğrusal-olmayan sistemlerin modellenmesinde kullanılabilen diğer bir araç ise Destek Vektör Makineleridir (SVM) (Vapnik 1995, Vapnik 1998[a,b]). İstatiksel Öğrenme Kuramı ve Yapısal Riski En Aza İndirme İlkesi'ne dayanan SVM algoritmaları herhangi bir sınıflandırma veya bağlanım problemini yerel minimumlara takılmadan çözebilir. Global minimumun bulunması, sınıflandırma veya bağlanım probleminin bir Karesel Programlama (Quadratic Programming-QP) problemine dönüştürülerek çözülmesiyle

sağlanır. Son on yılda, SVM tabanlı algoritmalar çok hızlı bir şekilde gelişmiş pek çok alana uygulanmıştır (Cristianini ve Taylor 2000).

Bu tezin amacı, çevrimiçi SVM-Tabanlı MPC yöntemini hem doğrusal sistemlere hem de doğrusal olmayan sistemlere uygulayarak sunmaktır. Tezin ikinci bölümünde, kısıtsız optimizasyon, üçüncü bölümünde kısıtlı optimizasyon, dördüncü bölümünde destek vektör makineleri, beşinci bölümde çevrimiçi destek vektör makineleri anlatılmıştır. Altıncı bölümde ise, SVM-Tabanlı MPC yapısı RBF çekirdeği için formülize edilmiştir, doğrusal ve doğrusal olmayan sistemler için ayrı ayrı benzetim sonuçları verilmiştir.

2. KISITSIZ OPTİMİZASYON

Herhangi bir kısıt içermeyen optimizasyon problemleridir. Alt bölümlerde, problemin tanımı ve optimizasyon probleminin çözümü için kullanılan yöntemler anlatılmıştır.

2.1 Bir-boyutlu Doğrusal-olmayan Nümerik Optimizasyon

Bir-boyutlu doğrusal-olmayan nümerik optimizasyon aşağıda standart biçimi verilen problemdeki gibi tek bir değişkenden oluşan bir fonksiyonun en aza indirgenmesini amaçlar.

2.1.1 Problemin Tanımı

Bir-boyutlu doğrusal-olmayan nümerik optimizasyon probleminin standart biçimi denklem (2.1)'de gösterilmiştir.

$$\begin{aligned} &\min_{x} f(x) \\ &x^l \leq x \leq x^u \end{aligned} \tag{2.1}$$

Burada *x* tasarım değişkenidir. *x* tasarım değişkeni belirtilen aralıkta öyle bir seçilmelidir ki, $f(x)$ fonksiyonu en küçük değerini alsın. Bu problemi çözmek için eldeki araçlara bağlı olarak çeşitli yaklaşımlar vardır.

İlk yaklaşım analitik yaklaşımdır. Buna göre $f(x)$ fonksiyonunun bir yerel minimumu, $f(x)$ fonksiyonunun türevinin alınıp sıfıra eşitleyerek elde edilen denklemi çözerek $f(x)$ fonksiyonunun ekstramum noktalarını yani minimum ve maksimum noktalarını bulup, ardından da bu noktaları $f(x)$ fonksiyonunun ikinci türevinde yerine koyarak bulunan noktanın minimum ya da maksimum noktası olduğu belirlenebilir. Bu çözüm matematiksel olarak şu şekilde ifade edilebilir: $f'(x^*) = 0$ denklemini sağlayan herhangi bir x^* ekstramum noktası $f''(x^*) > 0$

şartını da sağlıyorsa, o zaman x^* noktası $f(x)$ fonksiyonunun bir yerel minimumudur. Bu yaklaşımdaki ilk sorun bulunan yerel minimumun belirlenen aralıkta olmayabileceğidir. Analitik yaklaşımdaki diğer bir sorun ise, $f(x)$ fonksiyonunun ekstramum noktalarını bulurken karşılaşılan bir sorundur. $f'(x^*) = 0$ denklemi her zaman analitik yolla çözülemeyebilir. Örneğin $f(x) = xe^{-x} + \cos x$ gibi bir fonksiyonun ekstramum noktalarını bulurken $f'(x) = e^{-x} - xe^{-x} - \sin x = 0$ denkleminin analitik olarak çözülmesi gerekir ki bu mümkün değildir.

Görüldüğü gibi analitik yaklaşımın işe yaramadığı problemlerle karşılaşmak mümkündür. Özellikle pek çok gerçek dünya problemi analitik yöntemle çözülememektedir. Bu nedenle, analitik yöntemlere alternatif olarak nümerik yöntemler geliştirilmiştir. Analitik olarak çözülemeyen bir optimizasyon problemi için literatürde çok çeşitli nümerik yöntemler önerilmiştir. Bu yöntemlerden en yaygın kullanılanları alt kısımlarda anlatılmıştır.

2.1.2 Dolaylı Yöntemler

Bu yöntemler, minimumu dolaylı olarak yani fonksiyonunun türevinin sıfıra eşit olduğu noktaları bulmaya çalışırlar.

2.1.2.1 Newton-Raphson Yöntemi

Bir fonksiyonun minimum noktasını bulmak için kullanılan Newton-Raphson yöntemi dolaylı bir yöntemdir. Ayrıca Newton-Raphson yöntemi minimumu bulunacak olan amaç fonksiyonu $f(x)$'in birinci ve ikinci türevine ihtiyaç duymakta olup aşağıdaki özelliklere sahiptir:

- Geometrik bir temele sahiptir.
- Doğrusal olarak açılmış Taylor serilerini kullanır.
- İteratiftir.
- Karesel olarak yakınsar.

Newton-Raphson yöntemi, amaç fonksiyonu $f(x)$'in minimum noktasının $\frac{\partial f(x)}{\partial x} = f'(x) = 0$ şartını sağlaması gerektiğinden hareketle, bu şartı sağlayan noktaları bulmaya çalışır. Dolayısıyla optimizasyon problemi $f'(x) = 0$ şeklinde bir kök bulma problemine dönüşmüş olur. Newton-Raphson yöntemi, $f'(x) = 0$ şeklindeki problemi çözmek için birinci dereceden Taylor açılımını kullanır. Bu açılıma göre herhangi bir x_k değeri için eşitlik (2.2) yazılabilir.

$$f'(x_k + \Delta x_k) \cong f'(x_k) + f''(x_k)\Delta x_k \tag{2.2}$$

Buradan $f'(x_k + \Delta x_k) = 0$ yapılarak Δx_k çekilirse, eşitlik (2.3) elde edilir.

$$\Delta \mathbf{x}_k = -\frac{f'(\mathbf{x}_k)}{f''(\mathbf{x}_k)} \tag{2.3}$$

Newton-Raphson yöntemi çözüme iteratif yolla yaklaşır, yani her adımda aşağıdaki gibi bir algoritmayı kullanarak çözüme yaklaşmaya çalışır (Yang ve diğ. 2005).

Newton-Raphson Algoritması

i. Başlangıç olarak bir x_0 belirle.

ii. $\Delta x_k = -\frac{f'(x_k)}{f''(x_k)}$ değişimini hesapla.

iii. $x_{k+1} \leftarrow x_k + \Delta x_k$ kuralı ile güncelle.

iv. Eğer $f'(x_k + \Delta x_k) = 0$ şartı sağlanıyorsa algoritmayı sonlandır, sağlanmıyorsa Adım 2'ye git.

Buradaki güncellemenin etkin olabilmesi için $f''(x_k)$'in sıfır olmaması gerekir. Ama $f'(x)$ fonksiyonunun düz olduğu yerlerde Δx_k değişiminin büyük, dik olduğu yerlerde de küçük olması kaçınılmazdır. Newton-Raphson yönteminin etkin olabilmesi için iterasyonların düz yerlerden yani eğimin çok küçük olduğu yerlerden kaçınması gerekir. Bu ise metodun en ciddi problemidir.

2.1.2.2 İkiye Bölme Yöntemi

İkiye bölme yöntemi de, Newton-Rapson yöntemi gibi dolaylı bir yöntemdir, yani amaç fonksiyonu $f(x)$'in türevinin sıfıra eşit olduğu noktayı bulmaya çalışır. Bu yöntemdeki nümerik teknik, bir fonksiyonun bir kökünün bir pozitif ve bir negatif değeri arasında kaldığı düşüncesine dayanmaktadır. Bulunan çözüm aslında, fonksiyonun sıfırının da içinde bulunduğu aralıktır. Nihai çözüm bu aralığın toleransı çok küçük tutularak bulunur. Bu bir kök bulma algoritması olduğundan, optimizasyon sırasında minimize edilecek fonksiyonun türevine uygulanır. Böylece, amaç fonksiyonunun minimumunun bulunması, türevin sıfırının bulunmasına indirgenmiş olur. İkiye bölme yöntemini doğrudan amaç fonksiyonunun minimumunun bulunmasına uygulamak da mümkündür.

Metodu başlatmak için x_a ve x_b gibi iki başlangıç noktasına ihtiyaç vardır. $f'(x)$'in bu noktalardaki değerleri zıt işaretli olmalıdır. Bu durumda bu iki nokta arasında en az bir sıfırın bulunduğu varsayılır. Her bir iterasyon sırasında, sınırlarında x_a ve x_b noktalarının bulunduğu aralık ikiye bölünür öyle ki kalan kısmın uç noktaları yine zıt işaretlidir, yani kök hala kalan kısımdadır. Bu iteratif teknik aşağıdaki algoritma ile ifade edilebilir (Yang ve diğ. 2005).

İkiye Bölme Algoritması

i. Başlangıç olarak x_a ve x_b değerlerini belirle öyle ki $x_a < x_b$ olsun.
ii. $x_k = x_a + \frac{x_b - x_a}{2}$ noktasını hesapla.
iii. Eğer $f'(x_k) = 0$ veya $(x_b - x_a) < 10^{-4}$ şartı sağlanıyorsa algoritmayı sonlandır, sağlanmıyorsa ve eğer $f'(x_k)f'(x_a) > 0$ şartı sağlanıyor ise $x_a \leftarrow x_k$ yap, sağlanmıyorsa $x_b \leftarrow x_k$ yap.
iv. Adım 2'ye git.

2.1.3 Doğrudan Yöntemler

Bu yöntemler minimumu doğrudan bulurlar, en yaygın kullanılan yöntem olan Altın Bölme Yöntemi alt bölümde anlatılmıştır.

2.1.3.1 Altın Bölme Yöntemi

Altın bölme yöntemi aralık daraltma yöntemleri içinde en cazip olanıdır. Bu yöntem aralığı uçlardan aynı oranda daraltır. Aralık uçları altın oran denilen 0.61803 oranı ile daraltmaktadır. Bu oran estetik ve matematikte çok önemli bir yere sahiptir. Bu yöntemin uygulanması kolaydır. Çünkü minimize edilecek fonksiyonun şekil ve süreklilik özelliklerinden bağımsız olarak çalışır. En önemlisi de, çözüme belli bir toleransla ulaşmak için gerekli iterasyon sayısı önceden tahmin edilebilir.

Altın-Oran Algoritması

i. Sınırların alt (x_{alt}) ve üst (x_{ust}) değerlerini belirle.
$(\Delta x)_{son}$ değerini belirle.

$\tau = 0.38197$

Tolerans=$\varepsilon = \frac{(\Delta x)_{son}}{x_{ust} - x_{alt}}$

İterasyon sayısı= $N = -2.078 \ln \varepsilon$

$k \leftarrow 1$

ii. Aşağıdaki değerleri hesapla:
$x_1 \leftarrow (1-\tau)x_{alt} + \tau x_{ust}, \;\; f_1 = f(x_1)$

$x_2 \leftarrow (1-\tau)x_{ust} + \tau x_{alt}, \;\; f_2 = f(x_2)$

iii. Eğer $k < N$ şartı sağlanıyorsa,

ve eğer $f_2 < f_1$ şartı sağlanıyorsa,

$x_{alt} \leftarrow x_1, x_1 \leftarrow x_2, f_1 \leftarrow f_2$

$x_2 \leftarrow \tau x_{alt} + (1-\tau)x_{ust}, f_2 \leftarrow f(x_2)$

$k \leftarrow k + 1$

Adım 2'ye git.

ve eğer $f_1 < f_2$ şartı sağlanıyorsa,

$x_{ust} \leftarrow x_2, x_2 \leftarrow x_1, f_2 \leftarrow f_1$

$x_1 \leftarrow \tau x_{ust} + (1-\tau)x_{alt}, f_1 \leftarrow f(x_1)$

$k \leftarrow k + 1$

Adım 2'ye git.

2.1.4 Bir-boyutlu Nümerik Optimizasyonun Önemi

Bir-boyutlu nümerik optimizasyon, çok boyutlu nümerik optimizasyon probleminin çözümü esnasında adım-aralığının belirlenmesinde kullanılır. Çok-boyutlu nümerik optimizasyon probleminde genel güncelleme kuralı daha sonra da görüleceği gibi eşitlik (2.4)'te verilmiştir.

$$\mathbf{x}_{k+1} = \mathbf{x}_k + s\mathbf{p} \tag{2.4}$$

Burada **p** arama yönünü, *s* de adım-aralığını göstermektedir. Bu problemde güncelleme yapılırken önce uygun bir arama yönü belirlenir. Arama yönü belirlendikten sonra, uygun bir adım aralığının seçimi artık bir-boyutlu bir nümerik optimizasyon problemine dönüşmüştür.

2.2 Çok-boyutlu Doğrusal-olmayan Nümerik Optimizasyon

Çok boyutlu doğrusal-olmayan nümerik optimizasyon aşağıda standart biçimi verilen problemdeki gibi birden fazla değişkenden oluşan bir fonksiyonun en aza indirgenmesini amaçlar.

2.2.1 Problemin Tanımı

Çok-boyutlu doğrusal-olmayan nümerik optimizasyon probleminin standart biçimi eşitlik (2.5)'te gösterilmiştir.

$$\min_{x_1, x_2, \ldots, x_n} f(x_1, x_2, \ldots, x_n) \tag{2.5}$$

$$x_i^l \leq x_i \leq x_i^u$$

Burada $x_1, x_2, \ldots, x_n$ tasarım değişkenleridir. Aşağıdaki vektör notasyonu ile bu optimizasyon problemi daha sade bir şekilde yazılabilir.

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{2.6}$$

Böylece,

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) \\ &\mathbf{x}^l \leq \mathbf{x} \leq \mathbf{x}^u \end{aligned} \tag{2.7}$$

2.2.2 Genel Güncelleme Kuralı

Çok-boyutlu doğrusal-olmayan optimizasyon problemini nümerik olarak çözerken tasarım değişkenlerinden oluşan **x** vektörü her iterasyonda eşitlik (2.8)'deki genel güncelleme kuralıyla güncellenir.

$$\mathbf{x}_{k+1} = \mathbf{x}_k + s\mathbf{p} \tag{2.8}$$

Burada **p** arama yönü, *s* de adım aralığıdır. Her adımda, uygun arama yönü bulunduktan sonra bir de uygun bir adım aralığı bulunur. Adım aralığının bulunması tipik bir bir-boyutlu optimizasyon problemidir. Arama yönünün bulunması ise bir sonraki alt kısımda görüleceği gibi bazı matematiksel temellere dayanmaktadır.

2.2.3 Matematiksel Temeller

2.2.3.1 Gradyan, Hessian ve Jacobian Matrisleri

Bu alt-kısımda, çok değişkenli bir fonksiyonun belli bir noktada değerinin azalması için değişkenlerin hangi yönde değiştirilmesi konusu ele alınmıştır. İlk olarak $f(x)$ gibi bir-değişkenli problemi ele alalım. Tasarım değişkenindeki değişimlere bağlı olarak bu fonksiyondaki değişimi analiz edebilmek için birinci ve ikinci türevlere ihtiyaç vardır. Bir-değişkenli bir fonksiyonun birinci türevi,

$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{f(x + \Delta x) - f(x)}{\Delta x} = \lim_{\Delta x \to 0} \frac{\Delta f(x)}{\Delta x} \tag{2.9}$$

şeklindeyken, ikinci türevi eşitlik (2.10)'daki gibidir.

$$\frac{d^2 f(x)}{dx^2} = \frac{d}{dx}\left(\frac{df(x)}{dx}\right) = \lim_{\Delta x \to 0} \frac{\Delta(\frac{df(x)}{dx})}{\Delta x} \tag{2.10}$$

Burada, $\Delta(.)$ notasyonu sonlu/önemli bir değişimi gösterirken, $d(.)$ ve $\delta(.)$ notasyonları diferansiyel/küçük değişimleri göstermektedir.

Şimdi de, benzer şekilde, $f(x_1, x_2, \dots, x_n)$ gibi *n*-değişkenli bir fonksiyonu ele alalım. Artık kısmi türevler söz konusu olmaktadır. *n*-değişkenli bir fonksiyonun birinci mertebeden kısmi türevleri,

$$\begin{aligned}
\frac{\partial f(x_1, x_2, \dots, x_n)}{\partial x_1} &= \lim_{\Delta x_1 \to 0} \frac{f(x_1 + \Delta x_1, x_2, \dots, x_n) - f(x_1, x_2, \dots, x_n)}{\Delta x_1} \\
\frac{\partial f(x_1, x_2, \dots, x_n)}{\partial x_2} &= \lim_{\Delta x_2 \to 0} \frac{f(x_1, x_2 + \Delta x_2, \dots, x_n) - f(x_1, x_2, \dots, x_n)}{\Delta x_2} \\
&\vdots \\
\frac{\partial f(x_1, x_2, \dots, x_n)}{\partial x_n} &= \lim_{\Delta x_n \to 0} \frac{f(x_1, x_2, \dots, x_n + \Delta x_n) - f(x_1, x_2, \dots, x_n)}{\Delta x_n}
\end{aligned} \tag{2.11}$$

$f(x_1, x_2, \dots, x_n)$ fonksiyonundaki değişim değişkenlerdeki değişimlerden kaynaklanır. Cebir konularından da bilindiği gibi $f(x_1, x_2, \dots, x_n)$ fonksiyonundaki değişim x_1'deki diferansiyel değişim dx_1, x_2'deki diferansiyel değişim dx_2 ve bu şekilde devam ederken en sonunda x_n'deki diferansiyel değişim dx_n'nin bir sonucu olarak denklem (2.12)'deki gibidir.

$$f(x_1, x_2, \dots, x_n) = \frac{\partial f(x_1, x_2, \dots, x_n)}{\partial x_1} dx_1 + \frac{\partial f(x_1, x_2, \dots, x_n)}{\partial x_2} dx_2 + \cdots + \frac{\partial f(x_1, x_2, \dots, x_n)}{\partial x_n} dx_n \tag{2.12}$$

Bir değişkenli fonksiyonun türevi o fonksiyonun belli bir noktadaki eğimi ile ilişkiliydi. Çok-değişkenli bir $f(x_1, x_2, \dots, x_n)$ fonksiyonunun eğimi ise Gradyan vektörü ile gösterilir. $f(x_1, x_2, \dots, x_n)$ fonksiyonunun gradyanı eşitlik (2.13)'teki gibidir.

$$\nabla f(x_1, x_2, \dots, x_n) = \begin{bmatrix} \frac{\partial f(x_1, x_2, \dots, x_n)}{\partial x_1} \\ \frac{\partial f(x_1, x_2, \dots, x_n)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x_1, x_2, \dots, x_n)}{\partial x_n} \end{bmatrix} \tag{2.13}$$

Gradyan vektörünün en önemli özelliği, herhangi bir noktadaki Gradyan vektörünün, o fonksiyonun en büyük artım yönünü göstermesidir.

Çok-değişkenli bir fonksiyonun Hessian matrisi eşitlik (2.14)'te görülmektedir.

$$\nabla^2 f(x_1, x_2, \dots, x_n) = \begin{bmatrix} \frac{\partial^2 f(x_1, x_2, \dots, x_n)}{\partial x_1^2} & \frac{\partial^2 f(x_1, x_2, \dots, x_n)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x_1, x_2, \dots, x_n)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x_1, x_2, \dots, x_n)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x_1, x_2, \dots, x_n)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x_1, x_2, \dots, x_n)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f(x_1, x_2, \dots, x_n)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x_1, x_2, \dots, x_n)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x_1, x_2, \dots, x_n)}{\partial x_n^2} \end{bmatrix} \quad (2.14)$$

İleriki konularda sıkça karşılacak olan Jacobian matrisi ise n-değişkenli N adet fonksiyon için eşitlik (2.15)'teki gibidir.

$$\mathbf{J}(x_1, x_2, \dots, x_n) = \begin{bmatrix} \frac{\partial f_1(x_1, x_2, \dots, x_n)}{\partial x_1} & \frac{\partial f_1(x_1, x_2, \dots, x_n)}{\partial x_2} & \cdots & \frac{\partial f_1(x_1, x_2, \dots, x_n)}{\partial x_n} \\ \frac{\partial f_2(x_1, x_2, \dots, x_n)}{\partial x_1} & \frac{\partial f_2(x_1, x_2, \dots, x_n)}{\partial x_2} & \cdots & \frac{\partial f_2(x_1, x_2, \dots, x_n)}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_N(x_1, x_2, \dots, x_n)}{\partial x_1} & \frac{\partial f_N(x_1, x_2, \dots, x_n)}{\partial x_2} & \cdots & \frac{\partial f_N(x_1, x_2, \dots, x_n)}{\partial x_n} \end{bmatrix} \quad (2.15)$$

2.2.3.2 Taylor Teoremi ve Taylor Açılımı

Belli bir x_k noktasında bu fonksiyon $f(x_k)$ değerini almaktadır. Bu x_k noktasında küçük bir Δx_k değişimi ile bu fonksiyonun azalmasını sağlamak için fonksiyonun bu noktada nasıl davrandığını analiz etmek gerekmektedir. Bu analiz için Taylor açılımı uygun bir araçtır. Buna göre, fonksiyon bir x_k noktasında $f(x_k)$ değerini almaktayken $x_k + \Delta x_k$ noktasında hangi değeri alacağı aşağıdaki gibi Taylor açılımı ile belirlenebilir.

$$f(x_k + \Delta x_k) = f(x_k) + \frac{df(x)}{dx}\left|\Delta x_k + \frac{1}{2}\frac{d^2 f(x)}{dx^2}\right|(\Delta x_k)^2 + h.o.t \quad (2.16)$$

Bu açılımda genellikle üçüncü veya daha yüksek dereceden terimler ihmal edilir ve bu duruma göre ya birinci türevli ya da hem birinci hem de ikinci türevli

terimler kullanılır. Dolayısıyla, Taylor açılımı yaklaşık olarak denklem (2.17)'deki gibi yazılır.

$$f(x_k + \Delta x_k) \cong f(x_k) + \frac{df(x)}{dx}\left|\Delta x_k + \frac{1}{2}\frac{d^2 f(x)}{dx^2}\right|(\Delta x_k)^2 \tag{2.17}$$

Buradan $\Delta f(x_k) = f(x_k + \Delta x_k) - f(x_k)$ farkı denklem (2.18)'deki gibi bulunur.

$$\Delta f(x_k) = f(x_k + \Delta x_k) - f(x_k) \cong \frac{df(x)}{dx}\left|\Delta x_k + \frac{1}{2}\frac{d^2 f(x)}{dx^2}\right|(\Delta x_k)^2 \tag{2.18}$$

Burada eşitliğin sağ tarafındaki ilk terim birinci dereceden değişim, ikinci terim ise ikinci dereceden değişim olarak adlandırılmaktadır.

Şimdi de çok-değişkenli bir fonksiyonun Taylor açılımını yazalım. Çok-değişkenli fonksiyonun $(x_1 = x_{1k}, x_2 = x_{2k}, \dots, x_n = x_{nk})$ noktasındaki değeri bilindiğinde, fonksiyonun $(x_1 = x_{1k} + \Delta x_{1k}, x_2 = x_{2k} + \Delta x_{2k}, \dots, x_n = x_{nk} + \Delta x_{nk})$ noktasındaki değerini yaklaşık olarak bulmak için Taylor serileri kullanışlı bir yöntemdir. Çok-değişkenli bir fonksiyonun Taylor açılımı eşitlik (2.19)'daki gibidir.

$$\begin{aligned} f(x_{1k} + \Delta x_{1k}, x_{2k} + \Delta x_{2k}, \dots, x_{nk} + \Delta x_{nk}) &= f(x_{1k}, x_{2k}, \dots, x_{nk}) + \\ \frac{\partial f(x_1,x_2,\dots,x_n)}{\partial x_1}\Delta x_{1k} + \frac{\partial f(x_1,x_2,\dots,x_n)}{\partial x_2}\Delta x_{2k} + \cdots + \frac{\partial f(x_1,x_2,\dots,x_n)}{\partial x_n}\Delta x_{nk} + \\ \frac{1}{2}\frac{\partial^2 f(x_1,x_2,\dots,x_n)}{\partial x_1^2}(\Delta x_{1k})^2 + \frac{1}{2}\frac{\partial^2 f(x_1,x_2,\dots,x_n)}{\partial x_1 \partial x_2}\Delta x_{1k}\Delta x_{2k} + \cdots + \\ \frac{1}{2}\frac{\partial^2 f(x_1,x_2,\dots,x_n)}{\partial x_1 \partial x_n}\Delta x_{1k}\Delta x_{nk} + \frac{1}{2}\frac{\partial^2 f(x_1,x_2,\dots,x_n)}{\partial x_2 \partial x_1}\Delta x_{2k}\Delta x_{1k} + \cdots + \\ \frac{1}{2}\frac{\partial^2 f(x_1,x_2,\dots,x_n)}{\partial x_2 \partial x_n}\Delta x_{2k}\Delta x_{nk} + \cdots + \frac{1}{2}\frac{\partial^2 f(x_1,x_2,\dots,x_n)}{\partial x_n \partial x_1}\Delta x_{nk}\Delta x_{1k} + \cdots + \\ \frac{1}{2}\frac{\partial^2 f(x_1,x_2,\dots,x_n)}{\partial x_n^2}(\Delta x_{nk})^2 + h.o.t \end{aligned} \tag{2.19}$$

Bu açılım, vektör notasyonu ile eşitlik (2.20)'deki gibi ifade edilebilir.

$$f(\mathbf{x}_k + \Delta\mathbf{x}_k) = f(\mathbf{x}_k) + [\nabla f(\mathbf{x}_k)]^T \Delta\mathbf{x}_k + \frac{1}{2}[\Delta\mathbf{x}_k]^T \nabla^2 f(\mathbf{x}_k)\Delta\mathbf{x}_k + h.o.t \tag{2.20}$$

$$\mathbf{x}_k = \begin{bmatrix} x_{1k} \\ x_{2k} \\ \vdots \\ x_{nk} \end{bmatrix}, \qquad \Delta\mathbf{x}_k = \begin{bmatrix} \Delta x_{1k} \\ \Delta x_{2k} \\ \vdots \\ \Delta x_{nk} \end{bmatrix} \tag{2.21}$$

2.2.3.3 İniş Yönü

Belli bir $\mathbf{x}_k$ noktasında çok-değişkenli bir fonksiyonun değerinin çok küçük bir $\Delta\mathbf{x}_k$ değişimi ile azalması için gerekli koşula *iniş yönü koşulu* ve bu koşulu sağlayan değişim miktarına da *iniş yönü* denmektedir. İniş yönü, fonksiyon $\mathbf{x}_k$ noktasındayken hangi yönde çok küçük bir ilerleme yapılmalı ki fonksiyonun değeri azalsın sorusuna cevap vermektedir. Şimdi iniş yönü şartını bulalım. İlerleme miktarı $\|\Delta\mathbf{x}_k\|$ çok küçük olduğundan, Taylor açılımında sadece birinci-dereceden terimler alınıp diğerleri ihmal edilebilir.

$$f(\mathbf{x}_k + \Delta\mathbf{x}_k) \cong f(\mathbf{x}_k) + [\nabla f(\mathbf{x}_k)]^T \Delta\mathbf{x}_k \tag{2.22}$$

$\mathbf{x}_k$ noktasındaki $\Delta\mathbf{x}_k$ değişimi ile fonksiyonun değerinin azalması, yani $f(\mathbf{x}_k + \Delta\mathbf{x}_k) < f(\mathbf{x}_k)$ şartını sağlaması isteniyor. Taylor yaklaşıklığı kullanılırsa bu şart eşitlik (2.23)'e dönüşecektir.

$$f(\mathbf{x}_k + \Delta\mathbf{x}_k) \cong f(\mathbf{x}_k) + [\nabla f(\mathbf{x}_k)]^T \Delta\mathbf{x}_k < f(\mathbf{x}_k) \tag{2.23}$$

Gerekli sadeleştirmeler yapıldıktan sonra aşağıdaki gibi *İniş Yönü Şartı* elde edilir.

$$[\nabla f(\mathbf{x}_k)]^T \Delta\mathbf{x}_k < 0 \tag{2.24}$$

2.2.4 Optimallik için Analitik Koşullar

Analitik koşullar, kısıtsız problem için optimum çözümün bulunmasında kullanılacak olan gerek ve yeter koşullardır. Eşitlik (2.25)'teki problemi ele alalım.

$$\min_{\mathbf{x}} f(\mathbf{x})$$
$$\mathbf{x}^l \leq \mathbf{x} \leq \mathbf{x}^u \tag{2.25}$$

$\mathbf{x}^*$ noktası bu problem için bir çözüm adayı olsun. $\mathbf{x}^*$ noktasının bir yerel minimum olması için eşitlik (2.26)'daki şartı sağlaması gerekiyordu.

$$\forall \mathbf{x} \in \mathcal{S} \text{ öyle ki } \|\mathbf{x}^* - \mathbf{x}\| < \epsilon \rightarrow f(\mathbf{x}^*) \leq f(\mathbf{x}) \tag{2.26}$$

Burada $\mathcal{S}$ notasyonu $\mathbf{x}^l \leq \mathbf{x} \leq \mathbf{x}^u$ koşulunun sağlandığı bölgeyi göstermektedir. Benzer şekilde, eğer bir $\mathbf{x}^*$ noktası aşağıdaki şartı sağlarsa bu nokta kesin yerel minimumdur.

$$\forall \mathbf{x} \in \mathcal{S} \; ve \; \mathbf{x} \neq \mathbf{x}^* \text{ öyle ki } \|\mathbf{x}^* - \mathbf{x}\| < \epsilon \rightarrow f(\mathbf{x}^*) < f(\mathbf{x}) \tag{2.27}$$

Bir fonksiyonun bir yerel minimuma sahipken hiç global minimumu olmaması mümkündür. Hatta, bir fonksiyonun ne yerel ne de global minimumu olmayabilir, her ikisi birden olabilir, birden fazla yerel minimumu olabilir. Bir noktanın optimumluğunu belirlemek için gerekli daha pratik koşullara ihtiyaç vardır. Bunları elde etmek için $f(\mathbf{x})$ fonksiyonunun birinci ve ikinci dereceden türevlerinin mevcut ve $\mathbf{x}^*$ noktası civarında sürekli olduğunu varsayacağız.

2.2.4.1 Birinci-dereceden Koşullar

Varsayalım ki $\mathbf{x}^*$ noktası $f(\mathbf{x})$ fonksiyonunun yerel minimumu olsun. Fonksiyonunun $\mathbf{x}^*$ noktası civarındaki birinci dereceden Taylor açılımı eşitlik (2.28)'deki gibidir.

$$f(\mathbf{x}^* + \mathbf{p}) \cong f(\mathbf{x}^*) + [\nabla f(\mathbf{x}^*)]^T \mathbf{p} \tag{2.28}$$

Burada $\mathbf{p}$ herhangi bir vektörüdür ve bir ilerleme yönünü göstermektedir. Burada $\nabla f(\mathbf{x}^*) = \mathbf{0}$ olduğu gösterilecektir. Eğer $\mathbf{x}^*$ bir yerel minimum ise bu noktada artık olurlu bir iniş yönü bulunamaz, yani mümkün olan tüm $\mathbf{p}$ ilerleme

yönleri için $[\nabla f(\mathbf{x}^*)]^T \mathbf{p} \geq 0$ olmaktadır. Bu durumda $\mathbf{x}^*$ noktası bir yerel minimum ise eşitlik (2.29)'u sağlamaktadır ki bu koşullara *birinci-dereceden/gerek koşullar* adı verilmektedir.

$$\nabla f(\mathbf{x}^*) = \mathbf{0} \tag{2.29}$$

Bu koşulları sağlayan noktaya *durağan nokta (stationary point)* denir. Birinci-dereceden denmesinin sebebi ise koşullarda birinci dereceden türevlerin bulunmasıdır. Sadece gerek şartlar optimum noktanın bulunmasına yetmeyebilir. Gerek şart denmesinin sebebi, $\mathbf{x}^*$ noktasının bir yerel minimum olabilmesi için sağlanması gereken şartlar olduğu içindir. Birinci dereceden koşulların sağlanması, $\mathbf{x}^*$ noktasının bir yerel minimum olmasına yetmez, çünkü bu koşulları yerel minimumun yanısıra yerel maksimum veya bir semer noktası da sağlayabilir. Yerel minimumlar ancak ikinci dereceden koşulların sağlanmasıyla diğerlerinden ayırt edilebilir.

2.2.4.2 İkinci-dereceden Koşullar

İkinci-dereceden koşullar çoğunlukla yeter koşullar olarak bilinir. Tekrar Taylor açılımını ele alalım. Bu kez ikinci dereceden yaklaşıklıklar kullanılacaktır.

$$f(\mathbf{x}^* + \mathbf{p}) \cong f(\mathbf{x}^*) + [\nabla f(\mathbf{x}^*)]^T \mathbf{p} + \frac{1}{2} \mathbf{p}^T \nabla^2 f(\mathbf{x}^*) \mathbf{p} \tag{2.30}$$

Eğer $\mathbf{x}^*$ noktası gerek şartı ($\nabla f(\mathbf{x}^*) = \mathbf{0}$) sağlıyorsa, eşitliğin sağ tarafındaki ikinci terim sıfır olur eşitlik (2.31)'deki gibi ifade edilir.

$$f(\mathbf{x}^* + \mathbf{p}) \cong f(\mathbf{x}^*) + \frac{1}{2} \mathbf{p}^T \nabla^2 f(\mathbf{x}^*) \mathbf{p} \tag{2.31}$$

Bu durumda $\mathbf{x}^*$ noktasından herhangi bir $\mathbf{p}$ yönünde ilerlediğimizde fonksiyondaki değişim $\Delta f(\mathbf{x}^*)$ eşitlik (2.32)'deki gibi olmalıdır.

$$\nabla f(\mathbf{x}^*) \cong \frac{1}{2}\mathbf{p}^T \nabla^2 f(\mathbf{x}^*)\mathbf{p} \tag{2.32}$$

$\mathbf{x}^*$ noktasının yerel minimum olması için $\Delta f(\mathbf{x}^*)$'ın sıfırdan büyük veya sıfıra eşit olması göz önüne alınırsa, eşitlik (2.33)'deki şart elde edilir ki bu da ikinci-dereceden/yeter koşul olmaktadır.

$$\nabla f(\mathbf{x}^*) \cong \frac{1}{2}\mathbf{p}^T \nabla^2 f(\mathbf{x}^*)\mathbf{p} \geq 0 \tag{2.33}$$

Yeter koşulun sağlanması için $\nabla^2 f(\mathbf{x}^*)$ matrisinin pozitif yarı tanımlı olması gerekir. Benzer şekilde, bir $\mathbf{x}^*$ noktasının kesin yerel minimum olması için $\nabla f(\mathbf{x}^*) = \mathbf{0}$ olmalı ve $\nabla^2 f(\mathbf{x}^*)$ Hessian matrisi pozitif tanımlı olmalıdır. $\nabla^2 f(\mathbf{x}^*)$ Hessian matrisinin pozitif tanımlı olması için aşağıdaki üç seçenekten birini sağlaması yeterlidir.

- $\forall \mathbf{p}$ için $\mathbf{p}^T \nabla^2 f(\mathbf{x}^*)\mathbf{p} > 0$ olmalı veya
- $\nabla^2 f(\mathbf{x}^*)$ matrisinin tüm özdeğerleri pozitif olmalı veya
- $\nabla^2 f(\mathbf{x}^*)$ matrisinin kendisi de dahil olmak üzere tüm alt-kare-matrislerinin determinantları pozitif olmalı.

2.2.5 Gradyan Yöntemler

Gradyan yöntemler, kısıtsız optimizasyon problemini, minimumu bulunacak fonksiyonun türev bilgisini kullanarak çözmeye çalışırlar. Bunun için de aşağıdaki gibi Taylor açılımından yararlanırlar.

$$f(\mathbf{x}_k + \Delta\mathbf{x}_k) = f(\mathbf{x}_k) + [\nabla f(\mathbf{x}_k)]^T \Delta\mathbf{x}_k + \frac{1}{2}[\Delta\mathbf{x}_k]^T \nabla^2 f(\mathbf{x}_k)\Delta\mathbf{x}_k + h.o.t \tag{2.34}$$

Burada $\nabla f(\mathbf{x}_k)$ gradyan terimi birinci-dereceden, $\nabla^2 f(\mathbf{x}_k)$ terimi de ikinci dereceden türev bilgisi içerir.

2.2.5.1 Birinci-dereceden Yöntemler

Birinci dereceden yöntemler Taylor açılımında sadece birinci-dereceden türev bilgisini kullandıkları için bu ismi almışlardır. Bu yöntemleri uygulamak için fonksiyonun sadece Gradyan vektörünü bilmek yeterlidir. Bu alt kısımda birinci dereceden yöntemlerin başlıcaları ele alınmıştır.

2.2.5.1.1 Dik-İniş (Steepest-Descent SD) Yöntemi

Bir fonksiyonun bir noktadaki Gradyat vektörünün, fonksiyonun o noktadaki en büyük artım yönünü gösterdiği daha önce belirtilmişti. Dik-İniş yöntemi de buradan hareketle, her adımda Gradyan vektörünün ters yönünde hareket ederek fonksiyonu azaltma ilkesine dayanmaktadır (Yang ve diğ. 2005).

Dik-İniş Algoritması

i. Bir başlangıç noktası $(\mathbf{x}_0)$ ve maksimum iterasyon sayısı (N_{max}) belirle.
Sonlandırma kriterleri için ε_1, ε_2 ve ε_3 değerlerini belirle.
$k \leftarrow 0$

ii. $\mathbf{x}_k$ noktasındaki gradyan vektörünü $\nabla f(\mathbf{x}_k)$ hesapla.
İlerleme yönü olarak $\mathbf{p}_k = -\nabla f(\mathbf{x}_k)$ seç.
Bir boyutlu optimizasyon ile $f(\mathbf{x}_k + \mathrm{s}_k\mathbf{p}_k)$ değerini minimum yapan adım aralığı (s_k) bul.
$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathrm{s}_k\mathbf{p}_k$ kuralı ile güncellemeyi yap.
$k \leftarrow k + 1$

iii. Aşağıdaki şartlardan herhangi biri sağlanıyorsa algoritmayı bitir, sağlamıyorsa Adım 2'ye git.

C1: $k > N$ maksimum iterasyon sayısına ulaşıldı.
C2: $|\Delta f| = |f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)| < \varepsilon_1$ fonksiyon değişmiyor.
C3: $|\Delta x| = |\mathbf{x}_{k+1} - \mathbf{x}_k| < \varepsilon_2$ nokta değişmiyor.
C4: $\|\nabla f(\mathbf{x}_{k+1})\| < \varepsilon_3$ algoritma yerel minimuma yakınsadı.

Dik-iniş yöntemi hafızasızdır. Önceki ilerleme yönlerini dikkate almaz ki bu da algoritmanın yerel minimuma daha fazla adımda yakınsamasına yol açar. Bu algoritmaya alternatif olarak bir önceki yönü de dikkate alan Conjuge-Gradient yöntemi önerilmiştir.

2.2.5.1.2 Conjugate-Gradient Yöntemi

Dik-İniş algoritmasının değiştirilmiş bir şeklidir. Arama yönü Hessian matrisine göre eşleniktir. *n* değişkenli bir karesel problemi *n*'den daha az iterasyonda çözer (Yang ve diğ. 2005).

Conjugate-Gradient Algoritması

i. Bir başlangıç noktası $(\mathbf{x}_0)$ ve maksimum iterasyon sayısı (N_{max}) belirle.
Sonlandırma kriterleri için ε_1, ε_2 ve ε_3 değerlerini belirle.
$k \leftarrow 0$

ii. $\mathbf{x}_k$ noktasındaki gradyan vektörünü $\nabla f(\mathbf{x}_k)$ hesapla.
Eğer $k = 0$ ise ilerleme yönü olarak $\mathbf{p}_k = -\nabla f(\mathbf{x}_k)$ seç.
Eğer $k \neq 0$ ise ilerleme yönü olarak $\mathbf{p}_k = -\nabla f(\mathbf{x}_k) + \beta \mathbf{p}_{k-1}$ seç.
Burada $\beta = \frac{\nabla^T f(\mathbf{x}_k) \nabla f(\mathbf{x}_k)}{\nabla^T f(\mathbf{x}_{k-1}) \nabla f(\mathbf{x}_{k-1})}$ şeklindedir.
Bir boyutlu optimizasyon ile $f(\mathbf{x}_k + \mathrm{s}_k \mathbf{p}_k)$ değerini minimum yapan adım aralığı (s_k) bul.
$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathrm{s}_k \mathbf{p}_k$ kuralı ile güncelleme yap.
$k \leftarrow k + 1$

iii. Aşağıdaki şartlardan herhangi biri sağlanıyorsa algoritmayı bitir, sağlamıyorsa Adım 2'ye git.

C1: $k > N$ maksimum iterasyon sayısına ulaşıldı.
C2: $|\Delta f| = |f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)| < \varepsilon_1$ fonksiyon değişmiyor.
C3: $|\Delta x| = |\mathbf{x}_{k+1} - \mathbf{x}_k| < \varepsilon_2$ nokta değişmiyor.
C4: $\|\nabla f(\mathbf{x}_{k+1})\| < \varepsilon_3$ algoritma yerel minimuma yakınsadı.

2.2.6 İkinci-dereceden Yöntemler-Newton Yöntemi

İkinci-dereceden yöntemler, Taylor açılımında hem birinci hem de ikinci dereceden türev bilgisini kullanırlar. Bu yöntemleri uygulamak için fonksiyonun gradyan vektörünün yanısıra Hessian matrisini de kullanmak gerekir. Bu alt kısımda ikinci-dereceden yöntemlerin başlıcaları ele alınmıştır.

2.2.6.1 Newton Yöntemi

Newton yöntemi, *k.* iterasyonda, bir $\mathbf{x}_k$ noktasındayken uygun ilerleme yönü olan $\mathbf{p}_k$ yönünü bulurken aşağıdaki gibi ikinci dereceden Taylor yaklaşıklığını kullanır.

$$f(\mathbf{x}_k + \mathbf{p}_k) \cong f(\mathbf{x}_k) + [\nabla f(\mathbf{x}_k)]^T \mathbf{p}_k + \frac{1}{2}[\mathbf{p}_k]^T \nabla^2 f(\mathbf{x}_k)\mathbf{p}_k \tag{2.35}$$

Bu yaklaşıklıkla $f(\mathbf{x}_k + \mathbf{p}_k)$ fonksiyonu karesel bir fonksiyonla temsil edilmektedir. $f(\mathbf{x}_k + \mathbf{p}_k)$ fonksiyonunu $\mathbf{p}_k$ vektörüne göre optimize etmek için $\mathbf{p}_k$'ya göre türevi alınıp sıfıra eşitlenirse eşitlik (2.36) elde edilir.

$$\frac{\partial f(\mathbf{x}_k + \mathbf{p}_k)}{\partial \mathbf{p}_k} = \nabla f(\mathbf{x}_k) + \nabla^2 f(\mathbf{x}_k)\mathbf{p}_k = \mathbf{0} \tag{2.36}$$

Böylece, Newton yönteminde *k.* iterasyondaki ilerleme yönü $\mathbf{p}_k$, eşitlik (2.37)'deki doğrusal denklem sisteminin çözümünden elde edilir.

$$\nabla^2 f(\mathbf{x}_k)\mathbf{p}_k = -\nabla f(\mathbf{x}_k) \tag{2.37}$$

Bu denklem sisteminin çözümünden bulunan Newton yönü her seferinde $f(\mathbf{x}_k + \mathbf{p}_k)$ fonksiyonunu minimize etmeye çalışır. Bu yaklaşım, $\mathbf{x}_k$ noktasındaki ikinci dereceden Taylor açılımına dayanır. $\mathbf{x}_k$ noktasındaki bu açılım doğrusal olmayan $f(\mathbf{x}_k + \mathbf{p}_k)$ fonksiyonunu ne kadar iyi temsil ederse, bulunan $\mathbf{p}_k$ yönü o kadar uygun bir yön olacaktır.

2.2.6.2 Değiştirilmiş Newton Yöntemi

Dejenere durumlar dışında Newton yöntemi karesel bir yakınsama hızına sahiptir. Eğer Newton yöntemi yakınsarsa bu yakınsama durağan noktaya olur. Ancak Newton yöntemi bu haliyle nadiren kullanılır. Yöntemi daha güvenilir ve işlemsel olarak daha az karmaşık yapmak için bazı modifikasyonlar yapılmıştır. Newton yöntemi yakınsamayabilir veya yakınsasa bile bu bir yerel minimum olmayabilir. Newton yönteminin yakınsamasını ve hatta mevcut ise bir yerel minimuma yakınsamasını garanti etmek için bazı ilave stratejiler işin içine katılabilir. Bunun için benimsenen yaklaşım ise $\nabla^2 f(\mathbf{x}_k)\mathbf{p}_k = -\nabla f(\mathbf{x}_k)$ denkleminin çözümünden bulunan $\mathbf{p}_k$ yönünü genel güncelleme kuralı ile $\mathbf{x}_{k+1} = \mathbf{x}_k + s_k\mathbf{p}_k$ içinde kullanmaktır ki burada s_k adım aralığı $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$ olacak şekilde seçilir. Klasik Newton yönteminde adım aralığı her zaman $s_k = 1$ olmaktadır ve fonksiyonun azalmasını garanti etmemektedir.

Klasik Newton yönteminde ilerleme yönü $s_k > 0$ olmak üzere $f(\mathbf{x}_k + s_k\mathbf{p}_k) < f(\mathbf{x}_k)$ olacak şekilde seçilir. Bu ancak $\mathbf{p}_k$'nın bir iniş yönü olmasıyla mümkün olabilir, yani $[\nabla f(\mathbf{x}_k)]^T\mathbf{p}_k < 0$ olmalıdır. Bu iniş yönünün, Newton yönteminde nasıl garanti edilebileceğini bulmak için klasik Newton yöntemindeki ilerleme yönünün $\mathbf{p}_k = -[\nabla^2 f(\mathbf{x}_k)]^{-1}\nabla f(\mathbf{x}_k)$ olduğunu hatırlayalım. Eğer $\mathbf{p}_k$ iniş yönü olacaksa denklem (2.38) şartı sağlanmalıdır.

$$[\nabla f(\mathbf{x}_k)]^T\mathbf{p}_k = -[\nabla f(\mathbf{x}_k)]^T[\nabla^2 f(\mathbf{x}_k)]^T\nabla f(\mathbf{x}_k) < 0 \tag{2.38}$$

veya başka bir ifadeyle,

$$[\nabla f(\mathbf{x}_k)]^T[\nabla^2 f(\mathbf{x}_k)]^{-1}\nabla f(\mathbf{x}_k) > 0 \tag{2.39}$$

şartı sağlanmalıdır. Bu iniş yönü şartı ancak ve ancak $[\nabla^2 f(\mathbf{x}_k)]^{-1}$ matrisinin pozitif tanımlı olmasıyla mümkündür. $\nabla^2 f(\mathbf{x}_k)$ matrisinin pozitif tanımlı olması şartı $[\nabla^2 f(\mathbf{x}_k)]^T\mathbf{p}_k < 0$ şartından daha kuvvetli bir şarttır. Bunu daha iyi açıklamak için Taylor açılımına geri dönelim:

$$f(\mathbf{x}_k + \mathbf{p}_k) \cong f(\mathbf{x}_k) + [\nabla f(\mathbf{x}_k)]^T \mathbf{p}_k + \frac{1}{2}[\mathbf{p}_k]^T \nabla^2 f(\mathbf{x}_k)\mathbf{p}_k \quad (2.40)$$

Newton formülü bu karesel program şeklindeki bu yaklaşıklığın $\mathbf{p}_k$'ya göre türevinin alınıp sıfıra eşitlenmesinden bulunmuştu. Karesel bir fonksiyonun bir minimuma sahip olabilmesi için $\nabla^2 f(\mathbf{x}_k)$ matrisinin pozitif tanımlı olması gerekir. Eğer $\nabla^2 f(\mathbf{x}_k)$ matrisi pozitif tanımlı ise minimum noktası türevin sıfıra eşitlenmesinden bulunabilir. Eğer $\nabla^2 f(\mathbf{x}_k)$ matrisi pozitif tanımlı değilse, bu karesel fonksiyonun bir minimumu olmaz.

İterasyon sırasında $\nabla^2 f(\mathbf{x}_k)$ matrisi pozitif tanımlı olmazsa bu matrisi uygun bir pozitif tanımlı matris ile değiştirmek en çok başvurulan yollardan biridir. Bu şekilde ilerleme yönünün iniş yönü olması garanti edilir. Bu ilerleme yönü, amaç fonksiyonunun $f(\mathbf{x})$ karesel yaklaşıklığın minimize edilmesi yönündedir. $\nabla^2 f(\mathbf{x}_k)$ matrisi her zaman simetrik bir matristir ve simetrik matrislerin özdeğerleri her zaman reeldir. Eğer $\nabla^2 f(\mathbf{x}_k)$ matrisi pozitif tanımlı bir matris ise,

$$\nabla^2 f(\mathbf{x}_k) = \mathbf{L}\mathbf{D}\mathbf{L}^T \quad (2.41)$$

şeklinde faktörlere ayrılabilir ki burada **D** matrisi diagonali reel sayılardan oluşan bir matristir.

Eğer $\nabla^2 f(\mathbf{x}_k)$ matrisi pozitif tanımlı değilse o zaman **D** matrisinin diagonalinde $d_{ii} < 0$ şeklinde negatif bir eleman olacaktır. Bu durumda d_{ii} pozitif elemanla yer değiştirir. **D** matrisindeki bu değişiklik, $\nabla^2 f(\mathbf{x}_k)$ matrisinde

$$\nabla^2 f(\mathbf{x}_k) \leftarrow \nabla^2 f(\mathbf{x}_k) + \mathbf{E} \quad (2.42)$$

şeklinde bir değişikliğe karşı düşmektedir ki burada **E** matrisi diagonal bir matristir. Artık faktörizasyon

$$\nabla^2 f(\mathbf{x}_k) + \mathbf{E} = \mathbf{L}\mathbf{D}\mathbf{L}^T \quad (2.43)$$

haline gelir ve iniş yönü

$$[\mathbf{L}\mathbf{D}\mathbf{L}^T]\mathbf{p}_k = -\nabla f(\mathbf{x}_k) \tag{2.44}$$

denkleminin çözülmesiyle bulunur.

Eğer $\nabla^2 f(\mathbf{x}_k)$ matrisi pozitif tanımlı değilse kullanılabilecek başka bir yöntem ise *Birim Matris Ekleme* yöntemidir. Bu matrise uygun bir ekleme yapılarak $\nabla^2 f(\mathbf{x}_k) + \mu\mathbf{I}$ matrisinin pozitif tanımlı olması sağlanır. $\nabla^2 f(\mathbf{x}_k)$ matrisinin özdeğerleri $\{\lambda_1, \lambda_2, \dots, \lambda_n\}$ ve bunlara karşı düşen özvektörler de $\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_n\}$ olsun. Bu durumda, $\nabla^2 f(\mathbf{x}_k)\mathbf{v}_i = \lambda_i \mathbf{v}_i$ olduğundan,

$$[\nabla^2 f(\mathbf{x}_k) + \mu\mathbf{I}]\mathbf{v}_i = \nabla^2 f(\mathbf{x}_k)\mathbf{v}_i + \mu\mathbf{v}_i = \lambda_i\mathbf{v}_i + \mu\mathbf{v}_i = (\lambda_i + \mu)\mathbf{v}_i \tag{2.45}$$

yazılabilir ve görüleceği gibi $[\nabla^2 f(\mathbf{x}_k) + \mu\mathbf{I}]$ matrisinin özdeğerleri $(\lambda_i + \mu)$ şeklinde olup özvektörleri $\nabla^2 f(\mathbf{x}_k)$ matrisinin özvektörleriyle aynıdır. $[\nabla^2 f(\mathbf{x}_k) + \mu\mathbf{I}]$ matrisi, tüm i'ler için $(\lambda_i + \mu) > 0$ olacak şekilde μ değeri arttırılarak pozitif tanımlı hale getirilebilir. Böylece $[\nabla^2 f(\mathbf{x}_k) + \mu\mathbf{I}]$ matrisi pozitif tanımlı olur ve tersi alınabilir.

Bir $\mathbf{A}$ simetrik kare matrisin pozitif tanımlı olup olmadığını yani $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T$ şeklinde faktörlere ayrılıp ayrılamayacağını belirlemek için *Cholesky Faktörizasyonu* adı verilen bir yöntem kullanılmaktadır. Bu yönteme göre, bir $\mathbf{A}$ simetrik kare matrisi pozitif tanımlıysa $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T$ şeklinde yazılabilmelidir ki burada $\mathbf{L}$ bir alt üçgen matris, $\mathbf{D}$ ise diagonal elemanları kesin pozitif olan diagonal bir matristir. Pozitif tanımlı simetrik bir matrisin bu gösterilimine $\mathbf{L}\mathbf{D}\mathbf{L}^T$ faktörizasyonu denir. $\mathbf{D}$ matrisinin diagonal elemanları kesin pozitif olduğundan,

$$\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T = \mathbf{L}\mathbf{D}^{0.5}\mathbf{D}^{0.5}\mathbf{L}^T = \hat{\mathbf{L}}\hat{\mathbf{L}} = \mathbf{R}\mathbf{R}^T \tag{2.46}$$

şeklinde yazılabilir, burada $\hat{\mathbf{L}}$ genel alt üçgen ve $\mathbf{R}$ genel üst üçgen matristir. Bu faktörizasyon *Cholesky Faktörizasyonu* ve $\mathbf{R}$ matrisi *Cholesky Faktörü* olarak adlandırılır. $\mathbf{R}$ matrisi, $\mathbf{A}$ matrisinin karekökü gibi görülebilir. Cholesky faktörleri, aşağıdaki gibi eleman-eleman eşleşme yapılarak bulunabilir.

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} = \begin{bmatrix} r_{11} & & & \\ r_{21} & r_{22} & & \\ \vdots & \vdots & \ddots & \\ r_{n1} & r_{n2} & \dots & r_{nn} \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & \dots & r_{1n} \\ & r_{22} & \dots & r_{2n} \\ & & \ddots & \vdots \\ & & & r_{nn} \end{bmatrix} \tag{2.47}$$

İlk satırların ilk elemanları aşağıdaki gibi eşlenirse

$$a_{11} = r_{11}^2 \tag{2.48}$$

r_{11} elemanı bulunuri ardından ilk satırlar

$$a_{12} = r_{11}r_{12}, a_{13} = r_{11}r_{13}, \dots, a_{1n} = r_{11}r_{1n} \tag{2.49}$$

şeklinde eşlenerek **R** matrisinin ilk satırı bulunur. r_{22} elemanının bulunması için $r_{12} = r_{21}$ olduğundan

$$a_{22} = r_{12}^2 + r_{22}^2 \tag{2.50}$$

denkleminden yararlanılır ve ardından ikinci satırdaki elemanlar eşlenerek **R** matrisinin ikinci satırı bulunur ve bu işleme **R** matrisinin tamamı bulunana kadar devam edilir.

Değiştirilmiş Newton Algoritması

i. Bir başlangıç noktası $(\mathbf{x}_0)$ ve maksimum iterasyon sayısı (N_{max}) belirle.
   Sonlandırma kriterleri için ε_1, ε_2 ve ε_3 değerlerini belirle.
   $k \leftarrow 0$

ii. $\mathbf{x}_k$ noktasındaki gradyan vektörünü $\nabla f(\mathbf{x}_k)$ hesapla.
   $\mathbf{x}_k$ noktasındaki Hessian matrisini $\nabla^2 f(\mathbf{x}_k)$ hesapla.
   Eğer Hessian matrisi pozitif tanımlıysa ilerleme yönü olarak
   $\mathbf{p}_k = -[\nabla^2 f(\mathbf{x}_k)]^{-1} \nabla f(\mathbf{x}_k)$ seç.
   Eğer Hessian matrisi pozitif tanımlı değilse o zaman uygun bir matris ilavesi ile onu pozitif tanımlı hale getir ve ilerleme yönü olarak $\mathbf{p}_k = -[\nabla^2 f(\mathbf{x}_k) + \mu \mathbf{I}]^{-1} \nabla f(\mathbf{x}_k)$ seç.

Bir boyutlu optimizasyon ile $f(\mathbf{x}_k + s_k\mathbf{p}_k)$ değerini minimum yapan adım aralığı (s_k) bul.

$\mathbf{x}_{k+1} = \mathbf{x}_k + s_k\mathbf{p}_k$ kuralı ile güncelleme yap. $k \leftarrow k + 1$

iii. Aşağıdaki şartlardan herhangi biri sağlanıyorsa algoritmayı bitir, sağlamıyorsa Adım 2'ye git.

C1: $k > N$ maksimum iterasyon sayısına ulaşıldı.

C2: $|\Delta f| = |f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)| < \varepsilon_1$ fonksiyon değişmiyor.

C3: $|\Delta x| = |\mathbf{x}_{k+1} - \mathbf{x}_k| < \varepsilon_2$ nokta değişmiyor.

C4: $\|\nabla f(\mathbf{x}_{k+1})\| < \varepsilon_3$ algoritma yerel minimuma yakınsadı.

2.2.6.3 Newton-benzeri Yöntemler

Bu yöntemler gerçekte Newton yöntemi olmamakla birlikte çözüme yakınlaştıkça Newton yöntemine benzediklerinden dolayı bu ismi almışlardır. Bu yöntemlere Değişken Metrik Yöntemler (Variable Metric Methods-VMM) adı da verilmektedir. Çünkü ilerleme yönünün bulunmasında kullanılan ve başlangıçta genellikle birim matris seklinde seçilen matris (metrik) büyüklüğü her adımda güncellenir ve yerel minimuma yaklaştıkça bu metrik Hessian matrisine benzemeye başlar ve dolayısıyla da yöntem Newton yöntemine benzemeye başlar. VMM yöntemleri çözüme yaklaştıkça Newton yöntemine benzediklerinden bunlara *quasi-Newton* veya *Newton-like* yöntemler de denmektedir. Conjugate Gradient yönteminin Dik-İniş yönteminden üstünlüğü, bir önceki iterasyondaki yönün de dikkate alınmasından kaynaklanıyordu. VMM yöntemlerinde ise geçmişte kullanılan bütün yönlere ait bilgi *metrik* adı verilen $n \times n$'lik bir matriste tutulmaktadır. Arama yönünün bulunmasında kullanılan bu matris her iterasyonda güncellenmektedir. Bu matris için başlangıç olarak simetrik, pozitif tanımlı bir matris atanır. Bu genellikle birim matristir. Yöntemin yakınsaması için matrisin her iterasyonda bu özelliğini koruması gerekir.

2.2.6.3.1 Davidson-Fletcher-Powell (DFP) Yöntemi

Bu yöntemde, çözüme ulaşıldığında metrik Hessian matrisinin tersi olur (Nocedal ve Wright 1999).

DFP Algoritması

i. Bir başlangıç noktası $(\mathbf{x}_0)$, maksimum iterasyon sayısı (N_{max}) ve metrik'in ilk değerini $\mathbf{M}_0$ belirle.
Sonlandırma kriterleri için ε_1, ε_2 ve ε_3 değerlerini belirle.
$k \leftarrow 0$

ii. Adım 2: $\mathbf{x}_k$ noktasındaki gradyan vektörünü $\nabla f(\mathbf{x}_k)$ hesapla.
Eğer $\mathbf{M}_k$ matrisi pozitif tanımlı ise ilerleme yönü olarak $\mathbf{p}_k = -\mathbf{M}_k \nabla f(\mathbf{x}_k)$ seç.
Eğer $\mathbf{M}_k$ matrisi pozitif tanımlı değil ise o zaman uygun bir matris ilavesi ile onu pozitif tanımlı hale getir ve ilerleme yönü olarak $\mathbf{p}_k = -[\mathbf{M}_k + \mu \mathbf{I}]\nabla f(\mathbf{x}_k)$ seç.
Bir boyutlu optimizasyon ile $f(\mathbf{x}_k + \mathrm{s}_k \mathbf{p}_k)$ değerini minimum yapan adım aralığı (s_k) bul.
$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathrm{s}_k \mathbf{p}_k$ kuralı ile güncelleme yap.
$\Delta \mathbf{x} = \mathrm{s}_k \mathbf{p}_k$ ve $\mathbf{y} = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$ olmak üzere metriki $\mathbf{M}_{k+1} = \mathbf{M}_k + \frac{\Delta \mathbf{x}[\Delta \mathbf{x}]^T}{\Delta \mathbf{x}^T[\mathbf{y}]} - \frac{[\mathbf{M}_k \mathbf{y}][\mathbf{M}_k \mathbf{y}]^T}{\mathbf{y}^T[\mathbf{M}_k \mathbf{y}]}$ şeklinde güncelle, $k \leftarrow k + 1$

iii. Aşağıdaki şartlardan herhangi biri sağlanıyorsa algoritmayı bitir, sağlamıyorsa Adım 2'ye git.
C1: $k > N$ maksimum iterasyon sayısına ulaşıldı.
C2: $|\Delta f| = |f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)| < \varepsilon_1$ fonksiyon değişmiyor.
C3: $|\Delta \mathbf{x}| = |\mathbf{x}_{k+1} - \mathbf{x}_k| < \varepsilon_2$ fonksiyon değişmiyor.
C4: $\|\nabla f(\mathbf{x}_{k+1})\| < \varepsilon_3$ algoritma yerel minimuma yakınsadı.

2.2.6.3.2 Broydon-Fletcher-Goldfarb-Shanno (BFGS) Yöntemi

VMM yöntemlerinin en popüler olanıdır. DFP'den farkı metriğin güncellenmesi şeklidir. DFP'de metrik Hessian matrisinin tersine yakınsarken,

BFGS'de Hessian matrisinin kendisine yakınsar. BFGS daha doğrudan bir yöntemdir. Bu matris için başlangıç olarak simetrik, pozitif tanımlı bir matris atanır. Bu genellikle birim matristir. Yöntemin yakınsaması için matrisin her iterasyonda bu özelliğini koruması gerekir. Çözüme ulaşıldığında ise bu matris Hessian matrisine eşit olur (Nocedal ve Wright 1999).

BFGS Algoritması

i. Adım 1: Bir başlangıç noktası $(\mathbf{x}_0)$ ve maksimum iterasyon sayısı (N_{max}) ve metrik'in ilk değerini $\mathbf{M}_0$ belirle.
Sonlandırma kriterleri için ε_1, ε_2 ve ε_3 değerlerini belirle.
$k \leftarrow 0$

ii. Adım 2: $\mathbf{x}_k$ noktasındaki gradyan vektörünü $\nabla f(\mathbf{x}_k)$ hesapla.
Eğer $\mathbf{M}_k$ matrisi pozitif tanımlı ise ilerleme yönü olarak $\mathbf{p}_k = -\mathbf{M}_k^{-1}\nabla f(\mathbf{x}_k)$ seç.
Eğer $\mathbf{M}_k$ matrisi pozitif tanımlı değilse o zaman uygun bir matris ilavesi ile onu pozitif tanımlı hale getir ve ilerleme yönü olarak $\mathbf{p}_k = -[\mathbf{M}_k + \mu\mathbf{I}]^{-1}\nabla f(\mathbf{x}_k)$ seç.
Bir boyutlu optimizasyon ile $f(\mathbf{x}_k + s_k\mathbf{p}_k)$ değerini minimum yapan adım aralığını (s_k) bul.
$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathrm{s}_k\mathbf{p}_k$ kuralı ile güncelleme yap.
$\Delta\mathbf{x} = \mathrm{s}_k\mathbf{p}_k$ ve $\mathbf{y} = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$ olmak üzere metriki $\mathbf{M}_{k+1} = \mathbf{M}_k + \frac{\mathbf{y}\mathbf{y}^T}{\mathbf{y}^T\Delta\mathbf{x}} + \frac{[\nabla f(\mathbf{x}_k)][\nabla f(\mathbf{x}_k)]^T}{[\nabla f(\mathbf{x}_k)]^T\mathbf{p}_k}$ şeklinde güncelle.
$k \leftarrow k + 1$

iii. Adım 3: Aşağıdaki şartlardan herhangi biri sağlanıyorsa algoritmayı bitir, sağlamıyorsa Adım 2'ye git.
C1: $k > N$ maksimum iterasyon sayısına ulaşıldı.
C2: $|\Delta f| = |f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)| < \varepsilon_1$ fonksiyon değişmiyor.
C3: $|\Delta x| = |\mathbf{x}_{k+1} - \mathbf{x}_k| < \varepsilon_2$ fonksiyon değişmiyor.
C4: $\|\nabla f(\mathbf{x}_{k+1})\| < \varepsilon_3$ algoritma yerel minimuma yakınsadı.

2.2.6.4 İkinci-dereceden Yaklaşık Yöntemler

Bu yöntemler, sadece birinci dereceden türev bilgisi kullanarak Hessian matrisini belli bir yaklaşıklıkla elde edip ikinci dereceden bir yakınsama sağlamaya çalışırlar. Eğer minimize edilecek $f(\mathbf{x})$ fonksiyonu belli sayıda karelerin toplamı şeklindeyse, yani

$$f(\mathbf{x}) = e_1^2(\mathbf{x}) + e_2^2(\mathbf{x}) + \cdots + e_N^2(\mathbf{x}) = \sum_{i=1}^{N} e_i^2(\mathbf{x}) = \mathbf{e}^T(\mathbf{x})\mathbf{e}(\mathbf{x}) \tag{2.51}$$

Burada $\mathbf{e}(\mathbf{x}) = [e_1(\mathbf{x}) \;\; e_2(\mathbf{x}) \;\; \ldots \;\; e_N(\mathbf{x})]^T$ şeklinde bir vektördür, bu durumda $f(\mathbf{x})$ fonksiyonunun Gradyan vektörünün j^{inci} elemanı eşitlik (2.52)'deki gibi olmalıdır.

$$[\nabla f(\mathbf{x})]_j = \frac{\partial f(\mathbf{x})}{\partial x_j} = 2\sum_{k=1}^{N} e_k(\mathbf{x})\frac{\partial e_k(\mathbf{x})}{\partial x_j} \tag{2.52}$$

Buna göre Gradyan vektörü eşitlik (2.53)'teki gibi ifade edilir.

$$\nabla f(\mathbf{x}) = 2\mathbf{J}^T(\mathbf{x})\mathbf{e}(\mathbf{x}) \tag{2.53}$$

Burada $\mathbf{J}(\mathbf{x})$ matrisi aşağıdaki gibi Jacobian matrisidir.

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \frac{\partial e_1(\mathbf{x})}{\partial x_1} & \frac{\partial e_1(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial e_1(\mathbf{x})}{\partial x_n} \\ \frac{\partial e_2(\mathbf{x})}{\partial x_1} & \frac{\partial e_2(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial e_2(\mathbf{x})}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial e_N(\mathbf{x})}{\partial x_1} & \frac{\partial e_N(\mathbf{x})}{\partial x_2} & \cdots & \frac{\partial e_N(\mathbf{x})}{\partial x_n} \end{bmatrix} \tag{2.54}$$

Benzer şekilde Hessian matrisinin $i^{inci}j^{inci}$ elemanı eşitlik (2.55)'te gösterilmiştir.

$$[\nabla^2 f(\mathbf{x})]_{ij} = \frac{\partial^2 f(\mathbf{x})}{\partial x_i \partial x_j} = 2\sum_{k=1}^{N}\left\{\frac{\partial e_k(\mathbf{x})}{\partial x_i}\frac{\partial e_k(\mathbf{x})}{\partial x_j} + e_k(\mathbf{x})\frac{\partial^2 e_k(\mathbf{x})}{\partial x_i \partial x_j}\right\} \tag{2.55}$$

Jacobian matrisi kullanılarak,

$$\nabla^2 f(\mathbf{x}) = 2\mathbf{J}^T(\mathbf{x})\mathbf{J}(\mathbf{x}) + 2\mathbf{S}(\mathbf{x}) \tag{2.56}$$

şeklinde yazılabilir ki burada $\mathbf{S}(\mathbf{x})$ matrisinin $i^{inci}j^{inci}$ elemanı eşitlik (2.57) ile verilmektedir.

$$[\mathbf{S}(\mathbf{x})]_{ij} = \sum_{k=1}^{N} e_k(\mathbf{x})\frac{\partial^2 e_k(\mathbf{x})}{\partial x_i \partial x_j} \tag{2.57}$$

Eğer $\mathbf{S}(\mathbf{x})$ matrisinin elemanlarının yeterince küçük olduğu varsayılırsa, o zaman Hessian matrisi eşitlik (2.58) gibi yazılabilir.

$$\nabla^2 f(\mathbf{x}) \cong 2\mathbf{J}^T(\mathbf{x})\mathbf{J}(\mathbf{x}) \tag{2.58}$$

Buradan görüleceği gibi ikinci-dereceden türev bilgisi içeren Hessian matrisi, birinci dereceden türev bilgisi içeren Jacobian matrisi yardımıyla belli bir hata ile bulunabilir.

2.2.6.4.1 Gauss-Newton (GN) Yöntemi

Hatırlanacağı gibi Newton yönteminde ilerleme yönü $\mathbf{p}_k = -[\nabla^2 f(\mathbf{x}_k)]^{-1}\nabla f(\mathbf{x}_k)$ şeklindeydi. Eğer bu yönü bulmak için gerekli büyüklükleri Jacobian matrisi karşılıkları kullanılırsa eşitlik (2.59)'daki gibi bir ilerleme yönü bulunur. Bu yönü kullanan yönteme *Gauss-Newton* yöntemi denir (Nocedal ve Wright 1999).

$$\begin{aligned}\mathbf{p}_k &= -[\nabla^2 f(\mathbf{x}_k)]^{-1}\nabla f(\mathbf{x}_k) \\ &\cong -[2\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k)]^{-1}2\mathbf{J}^T(\mathbf{x}_k)\mathbf{e}(\mathbf{x}_k)\end{aligned} \tag{2.59}$$

$$\cong [\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k)]^{-1}\mathbf{J}^T(\mathbf{x}_k)\mathbf{e}(\mathbf{x}_k)$$

Ancak bu yöntem pratikte çok fazla tercih edilmez. Çünkü uygulama sırasında her iterasyonda $\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k)$ matrisinin tersinin alınabiliyor olması gerekir. Ancak zaman zaman bu matris tekil olabilmektedir ki bu durumda bu yöntem uygulanamaz hale gelir. Bu durumu ortadan kaldırmak için Levenberg-Marquardt (LM) yöntemi önerilmiştir.

2.2.6.4.2 Levenberg-Marquardt (LM) Yöntemi

Gauss-Newton yönteminde karşılaşılabilecek bir problem $\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k)$ matrisinin tersinin olmamasıdır. O yüzden, bu matrise $\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k) + \mu_k\mathbf{I}$ matrisi pozitif tanımlı olacak şekilde bir $\mu_k\mathbf{I}$ terimi ilave edilir ki bu durumda ilerleme yönü eşitlik (2.60)'daki gibi Levenberg-Marquardt (LM) yönüne dönüşür.

$$\nabla^2 f(\mathbf{x}_k) = \mathbf{L}\mathbf{D}\mathbf{L}^T \tag{2.60}$$

Burada dikkat edilirse Levenberg-Marquardt yönündeki μ_k büyüklüğü her iterasyonda değişmektedir. μ_k büyüklüğünün ayarlanması aşağıdaki Levenberg-Marquardt algoritması içerisinde gerçekleştirilir (Nocedal ve Wright 1999).

Levenberg-Marquardt Algoritması

i. Bir başlangıç noktası $(\mathbf{x}_0)$, μ_0 başlangıç değeri ve maksimum iterasyon sayısı (N_{max}) belirle.
   μ değerinin değişimi için $\mu_{scal} > 0$, μ_{max} ve μ_{min} belirle.
   Sonlandırma kriterleri için ε_1, ε_2 ve ε_3 değerlerini belirle.
   $k \leftarrow 0$

ii. $\mathbf{x}_k$ noktasındaki fonksiyon değerini $f(\mathbf{x}_k)$, hata vektörünü $\mathbf{e}(\mathbf{x}_k)$ ve Jacobian matrisini $\mathbf{J}(\mathbf{x}_k)$ hesapla.

iii. Aday ilerleme yönü $\mathbf{z}_k = -[\mathbf{J}^T(\mathbf{x}_k)\mathbf{J}(\mathbf{x}_k) + \mu_k\mathbf{I}]^{-1}\mathbf{J}^T(\mathbf{x}_k)\mathbf{e}(\mathbf{x}_k)$
   Eğer $f(\mathbf{x}_k + \mathbf{z}_k) < f(\mathbf{x}_k)$ ise güncelle: $\mathbf{p}_k \leftarrow \mathbf{z}_k$ ve $\mu_k \leftarrow \mu_k/\mu_{scal}$
   Adım 5'e git.

iv. $f(\mathbf{x}_k + \mathbf{z}_k) > f(\mathbf{x}_k)$ ise

$\mu_k \leftarrow \mu_k . \mu_{scal}$ yap.

Eğer $\mu_k < \mu_{max}$ ve $\mu_{min} < \mu_k$ ise Adım 3'e git, aksi halde Adım 5'e git.

v. Adım 5: $k \leftarrow k + 1$

vi. Adım 6: Aşağıdaki şartlardan herhangi biri sağlanıyorsa algoritmayı bitir, sağlanmıyorsa Adım 2'ye git.

C1: $k > N$ maksimum iterasyon sayısına ulaşıldı.

C2: $|\Delta f| = |f(\mathbf{x}_{k+1}) - f(\mathbf{x}_k)| < \varepsilon_1$ fonksiyon değişmiyor.

C3: $|\Delta x| = |\mathbf{x}_{k+1} - \mathbf{x}_k| < \varepsilon_2$ fonksiyon değişmiyor.

C4: $\|\nabla f(\mathbf{x}_{k+1})\| < \varepsilon_3$ algoritma yerel minimuma yakınsadı.

Bu algoritmadan da görüleceği gibi fonksiyonu azaltan uygun bir ilerleme yönü bulunduğunda μ_k büyüklüğü azaltılarak $\mathbf{p}_k$ ilerleme yönü Gauss-Newton yönüne dönüşür ki bu durumda yakınsama hızlanır. Diğer taraftan, fonksiyonu azaltan uygun bir ilerleme yönünün bulunmaması halinde, μ_k değeri fonksiyonu azaltan uygun bir ilerleme yönü bulunana kadar artırılır ve böylece $\mathbf{p}_k$ ilerleme yönü Dik-İniş yönüne benzemeye benzemeye başlar ki bu durumda yakınsamanın yavaşlaması pahasına da olsa fonksiyonun azalması sağlanmaya çalışılır. Sonuç olarak, LM algoritması, yavaş ama güvenilir Dik-İniş yönü ile hızlı ama az güvenilir Gauss-Newton yönü arasında uygun bir geçiş sağlar. Bu da LM algoritmasının en güçlü yanıdır.

3. KISITLI OPTİMİZASYON

Kısıtlı optimizasyonda, kısıtlayıcı fonksiyonlar optimum çözümün bulunmasında önemli rol oynarlar. Kısıtlı optimizasyon problemleri, kısıtlayıcının tipine bağlı olarak; eşitlik kısıtlayıcı ve eşitsizlik kısıtlayıcı olmak üzere ikiye ayrılır ve her iki durum için farklı yaklaşımlar optimum çözümü elde etmek için kullanılır. Optimum çözümün bulunması ise bir sonraki alt kısımda görüleceği gibi bazı matematiksel temellere dayanmaktadır.

3.1 Matematiksel Temeller

3.1.1 Null ve Range Uzayları

$\mathbf{A}$ matrisi $m \leq n$ olmak üzere $m \times n$ boyutlu bir matris olsun. $\mathbf{A}$ matrisinin boş uzayı (null space) $\mathcal{N}(\mathbf{A})$ ile gösterilir ve aşağıdaki gibi tanımlanır:

$$\mathcal{N}(\mathbf{A}) = \{\mathbf{p} \in \mathbb{R}^n : \mathbf{Ap} = \mathbf{0}\} \tag{3.1}$$

Burada $\mathbf{0}$ sıfırlardan oluşan $n \times 1$ boyutlu bir vektördür. Bir matrisin boş uzayı, o matrisin tüm satırlarına dik olan vektörlerden oluşan bir kümedir. $\mathbf{Ax} = \mathbf{b}$ şeklindeki kısıtlara sahip bir problemde boş-uzay tüm olurlu (feasible) yönleri temsil eder. $\mathcal{N}(\mathbf{A})$ uzayındaki iki vektörün doğrusal kombinasyonu yine $\mathcal{N}(\mathbf{A})$ içinde bir vektör olacakır, yani $\mathcal{N}(\mathbf{A})$ uzayı $\mathbb{R}^n$ uzayının bir alt uzayıdır. $\mathcal{N}(\mathbf{A})$ uzayının boyutu $n - \text{rank}(\mathbf{A})$ şeklindedir. $\mathbf{A}$ matrisi tam-satır-rankına sahipse o zaman $\mathcal{N}(\mathbf{A})$ uzayının boyutu $n - m$ olur.

Diğer bir uzay da Range uzayıdır. Bu uzay, $\mathbf{A}$ matrisinin sütunları tarafından taranmaktadır, yani, $\mathbf{A}$ matrisinin Range uzayı onun sütunlarının doğrusal bir kombinasyonudur. $\mathbf{A}^T$ matrisinin range uzayı aşağıdaki gibi tanımlanmaktadır:

$$\mathcal{R}(\mathbf{A}^T) = \{\mathbf{q} \in \mathbb{R}^n : \mathbf{q} = \mathbf{A}^T\boldsymbol{\lambda}, \quad \boldsymbol{\lambda} \in \mathbb{R}^m\} \tag{3.2}$$

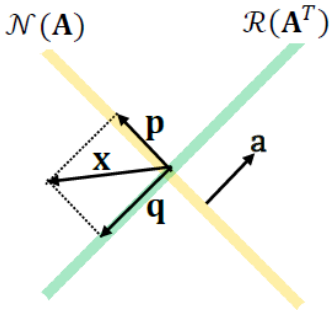
Range uzayının boyutu ile $\mathbf{A}^T$ matrisinin rankı aynıdır ve aynı zamanda $\mathbf{A}$ matrisinin rankına eşittir. $\mathcal{N}(\mathbf{A})$ ile $\mathcal{R}(\mathbf{A}^T)$ arasında önemli bir ilişki vardır: Bir matrisin boş uzayı ile transpozunun range uzayı birbirine diktir. Bu ifadeyi doğrulamak için, $\mathbf{q} \in \mathcal{R}(\mathbf{A}^T)$ vektörü herhangi bir $\boldsymbol{\lambda} \in \mathbb{R}^m$ için $\mathbf{q} = \mathbf{A}^T\boldsymbol{\lambda}$ şeklinde yazılabilir. Yani, $\mathbf{p} \in \mathcal{N}(\mathbf{A})$ ve $\mathbf{q} \in \mathcal{R}(\mathbf{A}^T)$ olmak üzere, aşağıdaki gibi yazılabilir:

$$\mathbf{q} = \mathbf{A}^T\boldsymbol{\lambda} \rightarrow q^T = \boldsymbol{\lambda}^T\mathbf{A} \rightarrow \mathbf{q}^T\mathbf{p} = \boldsymbol{\lambda}^T\mathbf{A}\mathbf{p} = \mathbf{0} \tag{3.3}$$

Daha da önemlisi, $\mathbf{A}$ matrisinin boş uzayı ile transpozunun range uzayı birbirine dik olduklarından ve boyutlarının toplamı $n - \text{rank}(\mathbf{A}) + \text{rank}(\mathbf{A}) = n$ olduğundan $\mathbf{x} \in \mathbb{R}^n$ şeklindeki herhangi bir vektör, $\mathbf{p}$ boş uzay bileşeni ve $\mathbf{q}$ da range uzay bileşeni olmak üzere,

$$\mathbf{x} = \mathbf{p} + \mathbf{q} \tag{3.4}$$

şeklinde yazılabilir. $\mathbf{A}$ matrisi, $\mathbf{A} = \mathbf{a}^\mathbf{T}$ şeklinde 1×2 boyutlu bir vektör olmak üzere, bu iki uzayın dikliği geometrik olarak Şekil 3-1'deki gibi gösterilebilir (İplikçi 2013). Burada dikkat edilirse, $\mathbf{a}$ vektörü boş uzaya diktir ve herhangi bir range-uzay vektörü $\lambda\mathbf{a}$ şeklinde ifade edilebilir.



Şekil 3-1: Boş uzay ve Range uzayın birbirine dikliği

Vektörleri, $m \times n$ boyutlu bir $\mathbf{A}$ matrisinin boş uzayında göstermek için aşağıdaki gibi $n \times r$ boyutlu bir $\mathbf{Z}$ boş uzay matrisi tanımlayalım:

$$\mathbf{AZ} = \mathbf{0} \tag{3.5}$$

Burada $\mathbf{0}$ sıfırladan oluşan $m \times r$ boyutlu bir matristir. Bu durumda, $\mathcal{N}(\mathbf{A})$ uzayındaki tüm vektörler $\mathbf{Z}$ boş uzay matrisinin sütunlarının doğrusal bir kombinasyonudur. $\mathbf{Z}$ boş uzay matrisinin gösterimi tek değildir. Eğer $\mathbf{A}$ matrisi tam satır rankına sahipse $(\text{rank}(\boldsymbol{A}) = m)$, o zaman $\mathbf{AZ} = \mathbf{0}$ şartını sağlayan ve rankı $n - m$ olan herhangi bir $n \times r$ boyutlu bir matris boş uzay matrisi olabilir. $\mathbf{Z}$ boş uzay matrisinin sütun sayısı r en az $n - m$ olmalıdır. $r = n - m$ durumunda $\mathbf{Z}$'nin sütunları doğrusal bağımsız olur ve $\mathbf{Z}$ boş uzay matrisi $\mathcal{N}(\mathbf{A})$ uzayının bir baz matrisi olur. $\mathcal{N}(\mathbf{A})$ uzayı şu şekilde ifade edilebilir:

$$\mathcal{N}(\mathbf{A}) = \{\mathbf{p}: \mathbf{p} = \mathbf{Zv}, \mathbf{v} \in \mathbb{R}^r\} \tag{3.6}$$

Böylece,

$$\mathcal{N}(\mathbf{A}) = \mathcal{R}(\mathbf{Z}) \tag{3.7}$$

yazılabilir. Bu bize olurlu yönlerin bulunmasında pratiklik sağlar. Örneğin, eğer bir $\hat{\mathbf{x}}$ noktası $\mathbf{A}\hat{\mathbf{x}} = \mathbf{b}$ kısıtlarını sağlayan olurlu bir nokta ise, o zaman diğer tüm olurlu noktalar aşağıdaki gibi yazılabilir.

$$\mathbf{x} = \hat{\mathbf{x}} + \mathbf{Zv}, \mathbf{v} \in \mathbb{R}^r \tag{3.8}$$

3.1.2 Doğrusal Kısıtların Gösterilimi

Burada amaç, olurlu bir noktadan, başka bir olurlu noktaya kolayca hareket edebilmek için kısıtların uygun bir formda yazılabilmesidir. Kısıtlar, değişkenler arasındaki ilişkileri belirlerler, örneğin bir değişkeni değiştirdiğimizde, olurluluk koşulları gereği diğer değişkenlerin de uygun şekilde değiştirilmesi gerekebilir. Kısıtların bir koordinat sistemi ile gösterilmesi daha kolaydır, böylece değişkenler arası ilişkilerin dikkate alınması ve olurlu noktalar arasındaki geçişler daha kolay hale gelir.

Genel durumda kısıtlar eşitlik ya da eşitsizlik biçimindedir. Doğrusal kısıtlı herhangi bir problem, aşağıdaki biçimde yazılabilir:

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) \\ &Kısıtlar: \mathbf{a}_i^T \mathbf{x} = b_i \;\; i \in \varepsilon \\ &\mathbf{a}_i^T \mathbf{x} \geq b_i \;\; i \in \mathcal{I} \end{aligned} \tag{3.9}$$

Burada hem eşitlik hem de eşitsizlik kısıtları görülmektedir. ε kümesinin elemanları eşitlik kısıtlarının indeksleri iken, $\mathcal{I}$ kümesinin elemanları eşitsizlik kısıtlarının indekslerinden oluşmaktadır. Bütün olurlu noktaların bulunduğu "olurlu bölge" $\mathcal{S}$ ile gösterilsin. Şimdi olurlu bir nokta ve onun yakın komşuluğundaki olurlu noktalara daha yakıdan bakalım. $\hat{\mathbf{x}}$ gibi olurlu bir noktadan bu noktanın yakınlarındaki başka bir olurlu noktaya hareket edildiğinde fonksiyonun nasıl değiştiğini inceleyeceğiz. İlk olarak hareketin yönüne bakalım. Eğer $\hat{\mathbf{x}}$ noktasındayken $\mathbf{p}$ yönünde küçük bir ilerleme yaptığımızda yine olurlu bir noktaya geliyorsak, o zaman $\mathbf{p}$ yönüne "olurlu yön (feasible direction)" denir. Başka bir deyişle, bir $\hat{\mathbf{x}}$ noktası tüm kısıtları sağlayan (olurlu) bir nokta olmak üzere $\hat{\mathbf{x}} + \mathbf{p}$ noktası da tüm kısıtları sağlıyorsa, o zaman $\mathbf{p}$ vektörüne "olurlu yön" diyeceğiz. Matematiksel olarak, $\hat{\mathbf{x}} \in \mathcal{S}$ iken $\hat{\mathbf{x}} + \alpha\mathbf{p} \in \mathcal{S}$ olacak şekilde küçük bir α bulunabiliyorsa, o zaman $\mathbf{p}$ olurlu bir yöndür. Böylece, olurlu bir yönden başka bir olurlu yöne ilerlemek olurluluğu korur. Pek çok uygulamada her iterasyonda olurluluğu korumak önemlidir. Örneğin, amaç fonksiyonu sadece olurlu noktalarda tanımlanmış olabilir veya problemin çözümü sadece olurlu noktalarda pratik bir anlama sahip olabilir.

Şimdi, olurlu yönleri, kısıt vektörler ($\boldsymbol{a}_i^T$'ler) cinsinden yazmaya çalışalım. Olurlu yönü tek bir kısıtlamayla karakterize etmekle başlayalım. Özellikle $\hat{\mathbf{x}}$ noktasındayken yapılan küçük bir yer değiştirme sonucu kısıtların sağlanmaya devam etmesi için gerekli koşulları belirleyelim. $\boldsymbol{a}_i^T \mathbf{x} = b_i$ gibi bir eşitlik kısıtı için olurlu yön $\boldsymbol{a}_i^T(\hat{\mathbf{x}} + \alpha\mathbf{p}) = b_i$ gibi olmalıdır, yani $\boldsymbol{a}_i^T \mathbf{p} = 0$ olmalıdır.

Benzer şekilde, $\boldsymbol{a}_i^T \mathbf{x} \geq b_i$ gibi bir eşitsizlik kısıtı için olurlu yön $\boldsymbol{a}_i^T(\hat{\mathbf{x}} + \alpha\mathbf{p}) \geq b_i$ gibi olmalıdır. Burada, aktif olmayan kısıt ($\boldsymbol{a}_i^T \hat{\boldsymbol{x}} \geq b_i$) için yeterince küçük

α değerlerinde $\boldsymbol{a}_i^T(\hat{\mathbf{x}} + \alpha\mathbf{p}) \geq b_i$ şartı zaten sağlanır. Diğer taraftan, aktif kısıt ($\boldsymbol{a}_i^T\hat{\mathbf{x}} = b_i$) içinse, yeterince küçük α değerlerinde $\boldsymbol{a}_i^T(\hat{\mathbf{x}} + \alpha\mathbf{p}) \geq b_i$ şartı ancak $\boldsymbol{a}_i^T\mathbf{p} \geq 0$ ile sağlanır.

Özet olarak, $\hat{\mathbf{x}}$ gibi olurlu bir noktadaki olurlu yönü sadece eşitlik kısıtları ve bu noktadaki aktif eşitsizlik kısıtları belirlemektedir. $\mathcal{I}$ kümesi, $\hat{\mathbf{x}}$ noktasında aktif olan eşitsizlik kısıtlarının indekslerinden oluşan küme olmak üzere $\hat{\mathbf{x}}$ noktasında olurlu $\mathbf{p}$ yönü denklem (3.10)'daki gibi tanımlanır.

$$\boldsymbol{a}_i^T\mathbf{p} = 0 \;\; i \in \varepsilon \cup \mathcal{I} \tag{3.10}$$

Eşitlik ve eşitsizlik kısıtlı problemleri ayrı ayrı ele almak yararlı olacaktır. Standart bir eşitlik kısıtlı problem denklem (3.11)'deki gibi olup olurlu yön şartı $\mathbf{Ap} = \mathbf{0}$ şeklindedir.

$$\begin{gathered}\min_{\mathbf{x}} f(\mathbf{x}) \\ Kısıtlar: \mathbf{Ax} = \mathbf{b}\end{gathered} \tag{3.11}$$

Diğer taraftan, standart bir eşitsizlik kısıtlı problem denklem (3.12)'deki ifade edilmektedir.

$$\begin{gathered}\min_{\mathbf{x}} f(\mathbf{x}) \\ Kısıtlar: \mathbf{Ax} \geq \mathbf{b}\end{gathered} \tag{3.12}$$

$\hat{\mathbf{x}}$ gibi olurlu bir noktada aktif olmayan kısıtların olurlu yönler üzerinde hiçbir etkisi yoktur. $\widehat{\mathbf{A}}$ matrisi, $\hat{\mathbf{x}}$ noktasında aktif olan kısıtların $\mathbf{A}$ martisinde karşılık gelen satırlarından oluşan bir alt matris olsun. Böylece, $\hat{\mathbf{x}}$ gibi olurlu noktada olurlu yön şartı $\widehat{\mathbf{A}}\mathbf{p} \geq \mathbf{0}$ şekline gelir.

Bir noktada aktif olmayan kısıtların olurlu yön üzerinde hiç bir etkisi olmadığından, bu noktanın optimal olup olmadığı test edilirken bu kısıtlar ihmal edilebilir. Özellikle optimum noktasında hangi kısıtın aktif olduğu bilinseydi aktif olmayan kısıtlar elenip aktif olanlar da sanki eşitlik kısıtıymış gibi düşünülerek

problem ele alınabilirdi. Eşitsizlik kısıtlı bir problemin bir çözümü aynı zamanda aktif kısıtlar tarafından tanımlanan eşitlik kısıtlı problemin de bir çözümüdür.

3.2 Kısıtlı Problemler için Optimallik Şartları

Bu kısımda, aşağıdaki biçimdeki problemler ele alınacaktır.

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) \\ &Kısıtlar: g_i(\mathbf{x}) = 0 \;\; i \in \varepsilon \\ &g_i(\mathbf{x}) \geq 0 \;\; i \in \mathcal{I} \end{aligned} \tag{3.13}$$

Burada ε eşitlik kısıtlarına ilişkin indeks kümesini, $\mathcal{I}$ de eşitsizlik kısıtlarına ilişkin indeks kümesini göstermektedir. Amaç fonksiyonu f ve kısıt fonksiyonları olan g_i'lerin ikinci türevlerinin mevcut olduğu varsayılmıştır. Bu kısımda, kısıtlı optimizasyon probleminin çözümü tarafından sağlanması gereken koşullar üzerinde çalışılacaktır. Kısıtsız durumda olduğu gibi sadece yerel çözümler üzerinde durulacak, global çözüm aranmayacaktır. Ancak, olurlu bölgenin ve f fonksiyonunun konveks olması durumunda yerel çözümün aynı zamanda global çözüm olduğu unutulmamalıdır.

Kısıtsız durumda optimallik koşulları, amaç fonksiyonunun yerel minimum $\mathbf{x}^*$civarındaki davranışını incelememize imkan sağlayan Taylor serisi yaklaşıklığı ile elde edilmişti. Kısıtsız durumda, özellikle $\mathbf{x}^*$ civarında amaç fonksiyonunun değeri azalmıyordu. Benzer bir yaklaşım kısıtlı bir durumda da kullanılabilir. Taylor serisi yaklaşıklığı ile, amaç fonksiyonu f ve kısıt fonksiyonları olan g_i'lerin $\mathbf{x}^*$ civarındaki davranışları analiz edilebilir. Bu durumda $\mathbf{x}^*$ civarındaki olurlu noktalarda amaç fonksiyonunun değerinin azalmaması gerektiği söylenebilir.

Burada optimallik koşulları aşama aşama çıkarılacaktır. Önce doğrusal kısıtlı problemler ele alınacak, ardından doğrusal olmayan kısıtlı problemlere geçilecektir. Her iki durumda da temel fikir aynı olmakla beraber doğrusal kısıtlı durumun analizi daha kolaydır.

Eğer tüm durumlar doğrusal ise, olurlu (feasible) hareketler tamamen olurlu yönlerle karakterize edilebilirler. $\mathbf{x}^*$ gibi bir minimum noktasında $f(\mathbf{x})$ amaç fonksiyonu için olurlu bir iniş yönü bulunamaz ve $\mathbf{x}^*$'daki tüm olurlu yönler için $\mathbf{p}^T \nabla f(\mathbf{x}^*) \geq 0$ şartı sağlanır. Birinci dereceden optimallik şartı da bunun bir sonucudur. Eğer doğrusal olmayan kısıtlar varsa, o zaman olurlu yönler boyunca $\mathbf{x}^*$'ın yakın noktalarında hareket etmek mümkün olmayabilir. Bu durumda hareketler olurlu yönler yerine olurlu eğriler üzerinde olacaktır. Olurlu bir eğri üzerindeki hareketlerin analizi, olurlu bir yön üzerindeki hareketlerin analizinden daha karmaşıktır. Yine de temel fikir, $\mathbf{x}^*$'ın yakın noktalarındaki olurlu noktalarda amaç fonksiyonunun değerinin azalmasıdır.

3.2.1 Doğrusal Kısıtlı Problemler için Optimallik Şartları

3.2.1.1 Doğrusal Eşitlik Kısıtlı Problemler için Optimallik Şartları

Aşağıdaki doğrusal eşitlik doğrusal problemi ele alalım:

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) \\ &Kısıtlar: \mathbf{a}_i^T \mathbf{x} = b_i \;\; i \in \varepsilon \end{aligned} \tag{3.14}$$

Satırları $\boldsymbol{a}_i^T$'lerden oluşan $m \times n$'lik matrisi $\mathbf{A}$ ve elemanları b_i'lerden oluşan $m \times n$'lik vektörü de $\mathbf{b}$ ile gösterelim. Böylece, kısıtları $\mathbf{Ax} = \mathbf{b}$ şeklinde göstermek mümkündür. Bu problemde şu varsayımlar yapılmaktadır:

1. Olurlu bölge içinde $f''(\mathbf{x})$ mevcuttur.
2. $\mathbf{A}$ matrisinin satırları doğrusal bağımsızdır.
3. Kısıtlar tutarlıdır.

Burada amaç, kısıtlı problemi kısıtsız probleme çevirmektir. $\mathbf{Ax} = \mathbf{b}$ şeklinde kısıtlara sahip bir problem, kısıtsız bir probleme dönüştürülebilir. Böylece kısıtsız problem için geliştirilen teori ve algoritmalar kısıtlı probleme de uygulanabilir.

$\mathbf{Ax} = \mathbf{b}$ şeklinde doğrusal eşitlik kısıtlarına sahip bir problem eşdeğer bir kısıtsız probleme dönüştürülebilir. Varsayalım ki $\hat{\mathbf{x}}$ olurlu bir nokta olsun yani $\mathbf{A}\hat{\mathbf{x}} = \mathbf{b}$ şartı sağlansın. Bu durumda başka bir olurlu nokta $\hat{\mathbf{x}} + \mathbf{p}$ ile ifade edilebilir ki burada $\mathbf{Ap} = \mathbf{0}$ olup olurlu bölge $\{\mathbf{x}: \mathbf{x} = \hat{\mathbf{x}} + \mathbf{p}, \mathbf{p} \in \mathcal{N}(\mathbf{A})\}$ şeklindedir. Bu olurlu bölgeyi boş uzay matrisi cinsinden de göstermek mümkündür. $n \times r$ boyutlu $\mathbf{Z}$ matrisi, $r \geq n - m$ olmak üzere $\mathbf{A}$'nın boş uzay matrisi olsun. $\mathbf{Z}$ matrisi aşağıdaki gibi r adet doğrusal bağımsız boş vektörlerden oluşan bir matristir.

$$\mathbf{Z} = [\mathbf{n}_1\ \mathbf{n}_2\ \dots\ \mathbf{n}_r] \tag{3.15}$$

Burada $i = 1, \dots, r$ için $\mathbf{An}_i = \mathbf{0}$ şeklindedir. Böylece olurlu bölge denklem (3.16)'daki gibi ifade edilebilir.

$$\{\mathbf{x}: \mathbf{x} = \hat{\mathbf{x}} + \mathbf{Zv}, \mathbf{v} \in \mathbb{R}^r\} \tag{3.16}$$

Sonuç olarak $\mathbf{x}$'e ilişkin doğrusal eşitlik kısıtlı problem, $\mathbf{v}$'e ilişkin kısıtsız probleme aşağıdaki gibi dönüştürülebilir:

$$\min_{\mathbf{v}} \emptyset(v) = f(\hat{\mathbf{x}} + \mathbf{Zv}) \tag{3.17}$$

Burada $\emptyset(.)$ fonksiyonu yeni amaç fonksiyonu olup orjinal amaç fonksiyonu olan $f(.)$'nin olurlu bölge üzerindeki izdüşümüdür ve "indirgenmiş fonksiyon (reduced function)" olarak adlandırılır. Eğer $\mathbf{Z}$ matrisi $\mathbf{A}$'nın boş uzayı için bir baz matrisi ise, yani sütunları baz vektörlerden oluşuyorsa, o zaman $\emptyset(.)$ fonksiyonunun değişken sayısı $n - m$ olur. Dolayısıyla, kısıtlı problem sadece kısıtsıza dönüşmez aynı zamanda değişken sayısı da azalır.

Optimallik koşullarını bulmak için indirgenmiş fonksiyonun gradyan vektörüne ve Hessian matrisine ihtiyaç vardır. İndirgenmiş gradyan vektörü, zincir kuralı ile denklem (3.18)'deki gibi ifade edilebilir.

$$\nabla\emptyset(\mathbf{v}) = \frac{\partial\emptyset(\mathbf{v})}{\partial\mathbf{v}} \tag{3.18}$$

$$= \nabla_{\mathbf{v}} \emptyset(\mathbf{v})$$

$$= \frac{\partial \mathbf{x}}{\partial \mathbf{v}} \frac{\partial f}{\partial \mathbf{x}}$$

$$= \mathbf{Z}^T \nabla_{\mathbf{x}} f(\mathbf{x})$$

$$= \mathbf{Z}^T \nabla_{\mathbf{x}} f(\hat{\mathbf{x}} + \mathbf{Z}\mathbf{v})$$

$$= \mathbf{Z}^T \nabla_{\mathbf{x}} f(\mathbf{x})$$

Benzer şekilde indirgenmiş Hessian matrisi de denklem (3.19)'daki gibidir.

$$\nabla^2 \emptyset(\mathbf{v}) = \nabla_{\mathbf{v}}^2 f(\hat{\mathbf{x}} + \mathbf{Z}\mathbf{v}) \tag{3.19}$$

Eğer $\mathbf{x}^*$ noktası kısıtlı problemin bir çözümü ise, o zaman $\mathbf{x}^* = \hat{\mathbf{x}} + \mathbf{Z}\mathbf{v}^*$ eşitliğini sağlayan her $\mathbf{v}^*$ noktası da indirgenmiş kısıtsız problemin bir çözümüdür. Böylece, $\mathbf{v}^*$ noktasında aşağıdaki gibi verilen optimallik koşulları sağlanmış olur:

$$\nabla \emptyset(\mathbf{v}^*) = 0 \; ve \; \nabla^2 \emptyset(\mathbf{v}^*) \; pozitif \; tanımlı \tag{3.20}$$

İndirgenmiş türev bilgilerini kullanarak birinci ve ikinci dereceden optimallik koşulları elde edilebilir. Yerel minimumunu bulmak istediğimiz doğrusal eşitlik kısıtlı probleme geri dönersek:

$$\min_{\mathbf{x}} f(\mathbf{x}) \qquad Kısıtlar: \mathbf{A}\mathbf{x} = \mathbf{b} \tag{3.21}$$

Eğer $\mathbf{x}^*$ noktası aşağıdaki yeter koşulları sağlıyorsa, o zaman $\mathbf{x}^*$ bir kesin yerel minimumdur:

- $\mathbf{A}\mathbf{x}^* = \mathbf{b}$
- $\mathbf{Z}^T \nabla f(\mathbf{x}^*) = 0$
- $\mathbf{Z}^T \nabla^2 f(\mathbf{x}^*)\mathbf{Z}$ matrisi pozitif tanımlı

Burada **Z** matrisi **A** matrisinin boş uzay matrisidir (İplikçi 2013).

3.2.1.1.1 Lagrangian Yaklaşımı

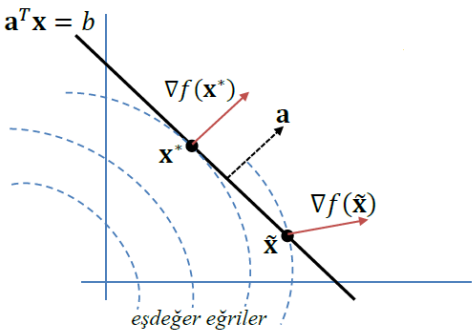
doğrusal eşitlik kısıtlı problemin çözüm noktasının sağlaması gereken birinci dereceden koşulun elde edilmesi için başka bir yaklaşım da Lagrangian yaklaşımıdır. Doğrusal eşitlik kısıtlı problemin bir yerel minimumu $\mathbf{x}^*$ ve $\mathbf{A}$ matrisinin de $\mathbf{Z}$ olsun. $\nabla f(\mathbf{x}^*)$ ifadesini boş uzay ve range uzay bileşenleriyle yazarsak:

$$\nabla f(\mathbf{x}^*) = \mathbf{Z}\mathbf{v}^* + \mathbf{A}^T\boldsymbol{\lambda}^* \tag{3.22}$$

Burada $\mathbf{v}^* \in \mathbb{R}^r$ ve $\boldsymbol{\lambda}^* \in \mathbb{R}^m$ şeklindedir. İfade üzerinden bazı işlemlerden sonra aşağıdaki eşitlik bulunur.

$$\nabla f(\mathbf{x}^*) = \mathbf{A}^T\boldsymbol{\lambda}^* \tag{3.23}$$

Yani, $\mathbf{x}^*$ noktasındaki Gradyan vektörü, kısıtların doğrusal kombinasyonudur. $\boldsymbol{\lambda}^*$ vektörü de bu doğrusal kombinasyonun katsayılarından oluşur ve "Lagrange çarpanları vektörü" olarak adlandırılır, i^{inci} elemanı i^{inci} kısıta karşı düşen Lagrange katsayısıdır. Bu, birinci dereceden optimallik koşuludur ve önceden bulunan ile eşdeğerdir. Optimallik koşulları Şekil 3-2'de gösterilmiştir (İplikçi 2013). Burada sadece $\mathbf{a}^T\mathbf{x} = b$ şeklinde tek bir kısıt vardır. Minimum noktası $\mathbf{x}^*$'daki Gradyan vektörü $\nabla f(\mathbf{x}^*)$, $\mathbf{a}$ vektörüne paraleldir ve dolayısıyla aralarında $\nabla f(\mathbf{x}^*) = \lambda^*\mathbf{a}$ ilişkisini sağlayan bir λ^* vardır. Diğer taraftan $\tilde{\mathbf{x}}$ gibi başka bir noktada Gradyan vektörü $\mathbf{a}$ vektörüne paralel değildir ve $\nabla f(\tilde{\mathbf{x}}) = \lambda^*\mathbf{a}$ ilişkisini sağlayan bir λ^* bulunamaz dolayısıyla da $\tilde{\mathbf{x}}$ optimal bir nokta değildir.



Şekil 3-2: Birinci dereceden optimallik koşulları

İndirgenmiş gradyan sıfırsa, yani $\mathbf{Z}^T\nabla f(\mathbf{x}^*) = \mathbf{0}$ ise, o zaman $\nabla f(\mathbf{x}^*) = \mathbf{A}^T\boldsymbol{\lambda}^*$ şartını sağlayan bir $\boldsymbol{\lambda}^*$ vektörü bulunabilir. Bunun tersi de doğrudur, yani eğer $\nabla f(\mathbf{x}^*) = \mathbf{A}^T\boldsymbol{\lambda}^*$ şartı sağlanıyorsa, o zaman indirgenmiş gradyan sıfır olur. Böylece, birinci dereceden optimallik koşulları eşdeğer bir şekilde denklem (3.24)'teki gibi yazılabilir.

$$\mathbf{Z}^T\nabla f(\mathbf{x}^*) = \mathbf{0} \Leftrightarrow \nabla f(\mathbf{x}^*) = \mathbf{A}^T\boldsymbol{\lambda}^* \tag{3.24}$$

Ancak, pratik açıdan bu iki eşdeğer koşul arasında bir fark vardır; eğer belli bir noktada indirgenmiş gradyan sıfır değilse, o zaman bu durum bir iniş yönü bulunmasında kullanılabilir. Diğer taraftan, eğer bir noktada Lagrange çarpanlarının bulunmamasının bir iniş yönü bulunmasında bir katkısı yoktur.

Kısıtlı problemde optimum çözüm bulunurken yapılan varsayım da **A** matrisinin tam satır rankına sahip olduğudur, yani tüm satırlarının doğrusal bağımsız olmasıdır. Bu varsayıma "regülerlik varsayımı" denmektedir. Bu kısımda geliştirilen yöntemler, regüler olmayan duruma da uyarlanabilir ama bu durumda Lagrange çarpanları genellikle tek olmamaktadır.

3.2.1.2 Lagrange Çarpanları ve Lagrange Fonksiyonu

Lagrange çarpanları, optimum noktasındaki gradyan vektörünü, kısıt matrisi olan **A** matrisinin doğrusal bir kombinasyonu şeklinde ifade ederler. Bu kısımda ayrıca, bu çarpanların, amaç fonksiyonunun optimum değerinin veriye olan duyarlılığını ifade etmekte de kullanıldığı görülecektir. Ayrıca, bu kısımda, Lagrange fonksiyonu tanıtılıp bunun optimallik koşullarının ifade edilmesinde nasıl kullanıldığı gösterilecektir.

Pek çok uygulamada, elde sadece yaklaşık veriler kullanılabilir durumda olmaktadır. Ölçüm hataları, verilerdeki dalgalanmalar ve bilginin yetersiz olması optimizasyon modelindeki belirsizliğe katkıda bulunan faktörlerdir. Kesin verilerin elde bulunmaması durumunda, en iyi tahminlerin kullanılmasından başka seçenek yoktur. Bir çözüm bulunduğunda, bir sonraki adım, bulunan çözümün kalitesinin

değerlendirilmesi olacaktır. Buradaki kritik soru şudur: bulunan çözüm verilerdeki değişime ne kadar duyarlıdır? Burada, kısıtların sağ tarafındaki verilerde küçük değişimler yapılarak bu değişimlerin optimum çözüm üzerindeki etkileri ele alınacaktır.

Aşağıdaki problemi ele alalım:

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) \\ &Kısıtlar: \mathbf{Ax} = \mathbf{b} \end{aligned} \tag{3.25}$$

Burada, $f(.)$ fonksiyonunun ikinci türevinin mevcut olduğu ve $m \times n$ $\mathbf{A}$ matrisinin tam satır rankına sahip olduğu varsayılmaktadır. Ayrıca, yerel minimum noktasının ($\mathbf{x}^*$) önceden bulunduğu varsayılmıştır. Şimdi, kısıtın sağ tarafında bulunan $\mathbf{b}$ vektörünün $\mathbf{b} + \boldsymbol{\delta}$ şeklinde çok küçük bir miktar pertübe edildiğini düşünelim. Amaç fonksiyonun optimum değerinin bu değişimden nasıl etkilendiğini inceleyelim. Eğer $\boldsymbol{\delta}$ vektöründeki değişimler yeterince küçükse, o zaman yeni optimum noktasının önceki optimum noktası $\mathbf{x}^*$'a yakın bir yerlerde olacağını söylemek yanlış olmaz. $\mathbf{x}^*$ noktasında ikinci dereceden koşullar sağlandığı sürece bunun doğruluğu ispatlanabilir. $\mathbf{x}^*$'a yakın bir nokta olan ve $\mathbf{A}\tilde{\mathbf{x}} = \mathbf{b} + \boldsymbol{\delta}$ kısıtını sağlayan $\tilde{\mathbf{x}}$ noktası için Taylor serisi yaklaşıklığı kullanılabilir.

$$\begin{aligned} f(\tilde{\mathbf{x}}) &\cong f(\mathbf{x}^*) + (\tilde{\mathbf{x}} - \mathbf{x}^*)^T \nabla f(\mathbf{x}^*) \\ &= f(\mathbf{x}^*) + (\tilde{\mathbf{x}} - \mathbf{x}^*)^T \mathbf{A}^T \boldsymbol{\lambda}^* \\ &= f(\mathbf{x}^*) + \boldsymbol{\delta}^T \boldsymbol{\lambda}^* \\ &= f(\mathbf{x}^*) + \sum_{i=1}^{m} \delta_i \lambda_i^* \end{aligned} \tag{3.26}$$

Özellikle, eğer $\tilde{\mathbf{x}}$ noktası pertübe edilmiş problemin bir optimum noktası ise bu eşitlikler geçerlidir. Eğer i^{inci} kısıtın sağ tarafı δ_i kadar değişirse, o zaman amaç fonksiyonu yaklaşık $\delta_i \lambda_i^*$ kadar değişir. Yani, λ_i^*, i^{inci} kısıtın sağ tarafındaki birim değişime karşı amaç fonksiyonundaki değişimi göstermektedir. Bu yüzden Lagrange çarpanlarına "ikincil değişkenler (dual variables)" denmektedir.

Optimallik koşullarına geri dönersek, herhangi bir çözümün olurlu olması gerektiğinden, bir yerel minimum noktası ($\mathbf{x}^*$) aşağıdaki gibi $m+n$ bilinmeyenli $m+n$ denklemin çözümüdür.

$$\begin{aligned} \nabla f(\mathbf{x}^*) - \mathbf{A}^T\boldsymbol{\lambda}^* &= \mathbf{0} \\ \mathbf{A}\mathbf{x}^* &= \mathbf{b} \end{aligned} \tag{3.27}$$

Bu denklem sistemi esasında birinci dereceden optimallik koşullarının başka bir gösterilimidir. Bu koşulları kullanan Lagrange'nın yaklaşımı ile, birinci dereceden optimallik koşullarını ifade etmek için aşağıdaki Lagrange fonksiyonundan yararlanılır.

$$\begin{aligned} \mathcal{L}(\mathbf{x},\boldsymbol{\lambda}) &= f(\mathbf{x}) - \sum_{i=1}^{m} \lambda_i(\mathbf{a}_i^T\mathbf{x} - b_i) \\ &= f(\mathbf{x}) - \boldsymbol{\lambda}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) \end{aligned} \tag{3.28}$$

Böylece birinci dereceden optimallik koşulları denklem (3.29)'daki gibi yazılabilir.

$$\nabla\mathcal{L}(\mathbf{x}^*,\boldsymbol{\lambda}^*) = \mathbf{0} \rightarrow \begin{cases} \nabla_{\mathbf{x}}\mathcal{L}(\mathbf{x}^*,\boldsymbol{\lambda}^*) = 0 \; sonucunda \; \nabla f(\mathbf{x}^*) = \mathbf{A}^T\boldsymbol{\lambda}^* \; elde \; edilir. \\ \nabla_{\boldsymbol{\lambda}}\mathcal{L}(\mathbf{x}^*,\boldsymbol{\lambda}^*) = 0 \; sonucunda \; \mathbf{A}\mathbf{x}^* = \mathbf{b} \; elde \; edilir. \end{cases} \tag{3.29}$$

Böylece, yerel minimum noktasının, Lagrange fonksiyonunun bir durağan noktası olduğu söylenebilir.

3.2.1.3 Doğrusal Eşitsizlik Kısıtlı Problemler için Optimallik Şartları

Aşağıdaki problemi ele alalım:

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) \\ &Kısıtlar: \mathbf{A}\mathbf{x} \geq \mathbf{b} \end{aligned} \tag{3.30}$$

Burada $\mathbf{A}$ satırları $\mathbf{a}_i^T$ şeklindeki vektörler olan $m \times n$ boyutlu bir matristir. Bu problemin olurlu bir çözümünün var olduğunu varsayıyoruz.

$\mathbf{x}^*$ noktası yerel bir çözüm olsun. $\mathbf{x}^*$ noktasında aktif olmayan kısıtların çözümünün optimalliği üzerinde hiç bir etkileri olmayacağından bu kısıtlar çıkartılabilir. $\hat{\mathbf{A}}$ matrisi, $\mathbf{A}$ matrisinin aktif kısıtlara denk düşen satırlarından oluşan bir matris, $\hat{\mathbf{b}}$ 'de aynı şekilde $\mathbf{b}$ vektörünün aktif kısıtlarına denk düşen elemanlarından oluşan bir vektör olsun. O zaman $\hat{\mathbf{A}}\mathbf{x}^* = \hat{\mathbf{b}}$ yazılabilir. $\mathbf{x}^*$ noktasından başka bir olurlu noktaya, $\mathbf{p}$ gibi olurlu bir iniş yönünde hareket ederek ulaşılabilir. $\mathbf{x}^*$ noktasında olurlu bir iniş yönü için koşul, $\hat{\mathbf{A}}\mathbf{p} \geq \mathbf{0}$ şeklindedir. Konuyu basitleştirmek için regülerlik varsayımı yapıyoruz, yani $\hat{\mathbf{A}}$ matrisinin satırlarının doğrusal bağımsız olduğunu varsayıyoruz.

$\mathbf{x}^*$ noktası eşitsizlik kısıtlı problemin bir yerel çözümü olduğundan aynı zamanda aşağıdaki eşitlik kısıtlı probleminde yerel çözümüdür.

$$\begin{aligned} &\min_{\mathbf{x}} f(\mathbf{x}) \\ &Kısıtlar: \hat{\mathbf{A}}\mathbf{x} = \hat{\mathbf{b}} \end{aligned} \tag{3.31}$$

$\mathbf{Z}$ matrisi, $\hat{\mathbf{A}}$ matrisinin boş uzay matrisi olsun. Eşitlik kısıtlı problemin birinci dereceden koşulu denklem (3.32)'deki gibidir.

$$\nabla f(\mathbf{x}^*) = \hat{\mathbf{A}}^T\hat{\boldsymbol{\lambda}}^* = \mathbf{0} \Leftrightarrow \mathbf{Z}^T\nabla f(\mathbf{x}^*) = \mathbf{0} \tag{3.32}$$

Burada $\hat{\boldsymbol{\lambda}}^*$ aktif kısıtlara denk düşen Lagrange çarpanları vektörüdür. İkinci dereceden gerek koşullar da $\mathbf{Z}^T\nabla^2 f(\mathbf{x}^*)\mathbf{Z}$ matrisinin pozitif yarı tanımlı olmasını gerektirir. Çözüm noktasında $\hat{\boldsymbol{\lambda}}^* \geq \mathbf{0}$ olmalıdır. Aksi halde $\hat{\boldsymbol{\lambda}}^*$ vektörünün bazı elemanları negatif olur ve çözüm noktasında hala olurlu bir iniş yönü bulunabilir ki bu da $\mathbf{x}^*$noktasının yerel minimum olduğu varsayımı ile çelişir. O yüzden çözüm noktasında $\hat{\boldsymbol{\lambda}}^* \geq \mathbf{0}$ koşulları sağlanmalıdır.

Birinci dereceden optimallik koşullarına bir de, aktif olmayan kısıtlara denk düşen Lagrange çarpanlarının sıfır olarak tanımlayarak, farklı açıdan bakalım. Bu

durumda tüm kısıtlara karşı düşen $m \times 1$ boyutlu bir $\hat{\boldsymbol{\lambda}}^*$ vektörü oluşturabiliriz. Böylece, $\nabla f(\mathbf{x}^*) = \hat{\mathbf{A}}^* \hat{\boldsymbol{\lambda}}^*$ ve $\hat{\boldsymbol{\lambda}}^* \geq 0$ koşulları, $\nabla f(\mathbf{x}^*) = \mathbf{A}^T \boldsymbol{\lambda}^*$ ve $\boldsymbol{\lambda}^* \geq 0$ koşullarına eşdeğerdir, burada $\mathbf{A}$ matrisinin aktif olmayan kısıtlara karşı düşen sütunları sıfır Lagrange çarpanı ile çarpılmaktadır. Aktif olmayan herhangi bir kısıta karşı sıfır Lagrange çarpanı olması gerekliliği denklem (3.33)'le ifade edilebilir.

$$\lambda_i^* \left(\mathbf{a}_i^T x^* - b_i \right) = 0, \qquad i = 1, \dots, m \tag{3.33}$$

Bu koşullar "tamamlayıcı gevşeklik koşulları (complementary slackness conditions)" olarak adlandırılır. Bu koşullara göre ya aktif kısıt olmalıdır ($\mathbf{a}_i^T x^* - b_i = 0$) ya da ona karşı düşen Lagrange çarpanı sıfır olmalıdır ($\lambda_i^* = 0$). Bu ikisinden en az biri sağlanmalıdır. Sadece birinin sağlandığı duruma ($\mathbf{a}_i^T x^* - b_i = 0 \; veya \; \lambda_i^* = 0$) "kesin tamamlayıcılık" denir. Bu durumda, aktif kısıta karşı düşen Lagrange çarpanları pozitifitir ($\hat{\boldsymbol{\lambda}}^* \geq \mathbf{0}$). Eğer kesin tamamlayıcılık şartı sağlanmıyorsa, o zaman bazı aktif kısıtların Lagrange çarpanları sıfır olacaktır ki bu tip kısıta *dejenere* denmektedir. Dejenere durum, istenen bir durum değildir çünkü algoritmanın yavaşlamasına hatta çökmesine bile yol açabilir.

Eşitsizlik kısıtlı problem için birinci ve ikinci dereceden gerek koşullar aşağıdaki şekilde özetlenebilir.

$$\begin{gathered} \min_{\mathbf{x}} f(\mathbf{x}) \\ Kısıtlar: \mathbf{A}\mathbf{x} \geq \mathbf{b} \end{gathered} \tag{3.34}$$

Eğer $\mathbf{x}^*$ noktası doğrusal eşitsizlik kısıtlı problemin bir yerel minimumu ise o zaman $\boldsymbol{\lambda}^*$ gibi bir Lagrange çarpanları vektörü için aşağıdaki koşullar sağlanır:

- $\nabla f(\mathbf{x}^*) = \mathbf{A}^T \boldsymbol{\lambda}^*$ veya eşdeğer olarak $\mathbf{Z}^T \nabla f(\mathbf{x}^*) = \mathbf{0}$
- $\boldsymbol{\lambda}^* \geq \mathbf{0}$
- $\boldsymbol{\lambda}^{*T} (\mathbf{A}\mathbf{x}^* - \mathbf{b}) = \mathbf{0}$
- $\mathbf{Z}^T \nabla^2 f(\mathbf{x}^*) \mathbf{Z}$ matrisinin pozitif yarı-tanımlı

Burada $\mathbf{Z}$ matrisi $\mathbf{x}^*$ noktasında aktif olan kısıtlara karşı düşen satırlardan oluşan matrisin bir boş uzay matrisidir.

Şimdi de, bir durağan noktanın gerçekten bir yerel minimum noktası olmasını garanti eden yeter koşulları elde etmek istiyoruz. Kolaylık olsun diye sadece $\mathbf{Z}$ matrisinin $\widehat{\mathbf{A}}$ matrisinin boş uzay matrisi olduğu durumu ele alacağız. Yeter koşul olarak, eşitlik kısıtlı problemde olduğu gibi $\mathbf{Z}^T\nabla^2 f(\mathbf{x}^*)\mathbf{Z}$ matrisinin pozitif tanımlı olması gerekeceği beklenmektedir. Doğrusal eşitsizlik kısıtlı problem için yeter koşullar şu şekilde ön teori ile yazılabilir.

$$\min_{\mathbf{x}} f(\mathbf{x}) \qquad Kısıtlar: \mathbf{Ax} \geq \mathbf{b} \tag{3.35}$$

Eğer $\mathbf{x}^*$ noktası aşağıdaki koşulları sağlarsa o zaman bu nokta doğrusal eşitsizlik kısıtlı problemin kesin yerel minimumudur.

- $\mathbf{Ax}^* \geq \mathbf{b}$
- $\nabla f(\mathbf{x}^*) = \mathbf{A}^T\boldsymbol{\lambda}^*$
- $\boldsymbol{\lambda}^* \geq \mathbf{0}$
- Kesin-tamamlayıcılık koşulu sağlanıyor.
- $\mathbf{Z}^T\nabla^2 f(\mathbf{x}^*)\mathbf{Z}$ matrisi pozitif tanımlı

Bu ön teoriyi, $\mathbf{x}^*$ noktasında herhangi bir olurlu yön boyunca amaç fonksiyonunun arttırdığını göstererek ispatlayabiliriz. Dikkat edilirse $\mathbf{x}^*$ noktası, üzerinde aktif olan kısıtlar için tanımlı $\{\mathbf{x}: \widehat{\mathbf{A}}\mathbf{x} = \hat{\mathbf{b}}\}$ kümesi için $f(.)$ amaç fonksiyonunun kesin yerel minimumudur. O yüzden $\widehat{\mathbf{A}}\mathbf{p} = \mathbf{0}$ koşulunu sağlayan herhangi bir $\mathbf{p}$ yönünde ilerlendiğinde $f(.)$ amaç fonksiyonu artacaktır. Şimdi de $\widehat{\mathbf{A}}\mathbf{p} \geq \mathbf{0}$ koşulunu sağlayan herhangi bir $\mathbf{p}$ yönünde ilerlemeyi ele alalım, ki burada $\widehat{\mathbf{A}}\mathbf{p}$'nin bazı elemanları pozitif olsun. $\mathbf{p}$ yönü olurlu bölgenin içine doğru bir yöndür. $\nabla f(\mathbf{x}^*) = \mathbf{A}^T\boldsymbol{\lambda}^* = \widehat{\mathbf{A}}^*\hat{\boldsymbol{\lambda}}^*$ olduğundan $\mathbf{p}^T\nabla f(\mathbf{x}^*) = \mathbf{p}^T\widehat{\mathbf{A}}^T\hat{\boldsymbol{\lambda}}^* > 0$ olur ve $\mathbf{p}$ yönü aynı zamanda bir olurlu artma yönü olur. Böylece, $\mathbf{x}^*$ noktasında herhangi bir olurlu yön boyunca ilerlendiğinde amaç fonksiyonunun değeri artar.

Görüldüğü gibi, aktif kısıta ilişkin Lagrange çarpanı eğer pozitifse, o zaman bu kısıttan olurlu bölgenin içine doğru yapılan küçük bir ilerlemede amaç fonksiyonunun değeri artar. Ancak, eğer Lagrange çarpanı sıfırsa, o zaman bu kısıttan olurlu bölgenin içine doğru yapılan küçük bir ilerlemede amaç fonksiyonunun değerinin artıp artmayacağı birinci dereceden bilgi ile bilinemez. O yüzden dejenere kısıt durumunda, bir noktanın yerel minimumluğunu garanti etmek için Hessian matrisi üzerinde daha sıkı koşulların sağlanması gerekecektir.

Aşağıdaki doğrusal eşitsizlik kısıtlı problemi ele alalım.

$$\min_{\mathbf{x}} f(\mathbf{x}) \qquad Kısıtlar: \mathbf{Ax} \geq \mathbf{b} \tag{3.36}$$

$\widehat{\mathbf{A}}_+$ matrisi, $\mathbf{A}$ matrisinin $\mathbf{x}^*$ noktasında dejenere olmayan aktif kısıtlara (yani $\lambda_i > 0$ olanlar) karşı düşen satırlarından oluşan bir alt matris olsun. $\mathbf{Z}_+$ matrisi de $\widehat{\mathbf{A}}_+$ matrisinin boş uzayı için bir baz matris olsun. Eğer $\mathbf{x}^*$ noktası aşağıdaki koşulları sağlarsa, o zaman bu nokta doğrusal eşitsizlik kısıtlı problemin kesin yerel minimumudur.

- $\mathbf{Ax}^* \geq \mathbf{b}$
- $\nabla f(\mathbf{x}^*) = \mathbf{A}^T \boldsymbol{\lambda}^*$
- $\boldsymbol{\lambda}^* \geq \mathbf{0}$
- $\boldsymbol{\lambda}^{*T}(\mathbf{Ax}^* - \mathbf{b}) = \mathbf{0}$
- $\mathbf{Z}_+^T \nabla^2 f(\mathbf{x}^*) \mathbf{Z}_+$ matrisi pozitif tanımlı

3.3 Duallik

Duallik (duality) kavramı hem doğrusal programlama hem de doğrusal olmayan programlama problemlerinde karşımıza çıkmaktadır. Bir doğrusal olmayan programlama problemine primal problem dersek, primal problem ile bunun duali arasında çok yakın ilişkiler vardır (İplikçi 2013).

- Eğer primal problem bir minimizasyon problemi ise, duali de bir maksimizasyon problemidir.
- Dual problemin duali yine primal problemdir.
- *Zayıf Duallik Problemi:* Maksimizasyon probleminin olurlu bölge içindeki bir çözümü, minimizasyon probleminin olurlu bölge içindeki bir çözümünün alt sınırıdır.
- *Kuvvetli Duallik Teoremi:* Eğer birinin (primal veya dual) optimal çözümü var ise ötekinin de vardır. Bu optimum çözümler birbirine eşittir.

Primal problem ile dual problem arasındaki ilişki hem teorik hem de işlemsel anlamda önemlidir. Örneğin dual problemi çözmek daha kolay olabilir ve dual problemin optimum çözümü biliniyorsa, primal problemin de optimum çözümü bulunabilir. Dual problemin optimum çözümüne ilişkin iyi bir tahmin, primal problemin optimum çözümüne ilişkin iyi bir tahminde bulunmasına yardımcı olabilir.

Birinci dereceden optimallik koşulları sadece tasarım değişkenleri (x_i'ler) için değil aynı zamanda da Lagrange çarpanları (λ_i'ler) için de yazılır. Lagrange çarpanları dual değişkenlerdir. Eğer optimum Lagrange çarpanları önceden bilinebilirse, optimizasyon probleminin çözümü çok kolaylaşacaktır. Dolayısıyla, optimum Lagrange çarpanlarının (λ_i^*'ler) kolayca hesaplanabilmesi oldukça faydalı olacaktır. Böylece soru şu hale gelmektedir: Bilinmeyen tasarım değişkenlerinin Lagrange çarpanları olduğu ve çözümün λ_i^*'lerden oluştuğu yeni bir doğrusal olmayan programlama problemi tanımlayabilir miyiz? Diğer bir soru da şudur: Bulunan bu çözüm hangi şartlar altında orjinal problemin çözümünde kullanılmaktadır? Duallik teorisi bu sorulara cevap aramaktadır.

3.3.1 Oyunlar ve Min-Max Dualliği

P ve D isimli iki oyuncu arasında geçen bir oyunda, P'nin elinde belli stratejilerden oluşan bir *X* kümesi olsun. Benzer şekilde, D'nin elinde de *Y* gibi bir stratejiler kümesi olsun. Oyun başladığında, P oyuncusu *X* kümesinden *x* gibi bir

strateji, D oyuncusu da Y kümesinden y gibi bir strateji belirlesin. Sonra, her ikiside seçtikleri stratejiyi aynı anda açıklasınlar. Sonuç olarak, P oyuncusu D oyuncusuna $F(x,y)$ miktarında para ödesin. Bu oyunun ismi *sıfır toplamlı oyun*'dur, çünkü oyunculardan birinin kaybettiği miktarı öteki kazanmaktadır.

Varsayalım ki hem P hem de D kazançlarını maksimize etmek için akılcı stratejiler seçmekte olsunlar. Yine varsayalım ki oyuncular kazançlarını maksimum yapacak stratejiyi belirlerken birbirlerinden etkilemiyor olsunlar. Böylece, her iki oyuncu da en kötü durum senaryolarını optimize etmeye çalışmaktadırlar.

İlk olarak P'yi ele alalım. Eğer P, $x \in X$ stratejisini seçerse, en kötü durumda (D çok zeki veya çok şanslı) D'ye ödeyeceği miktar,

$$F^*(x) = \max_{y \in Y} F(x,y) \tag{3.37}$$

olacaktır. Bu en kötü durum zararını minimize etmek için P,

$$\min_{x \in X} F^*(x) \tag{3.38}$$

şeklindeki problemi çözerek stratejiyi belirlemek zorundadır. Bu problem esasında bir min-max problemidir, çünkü problem

$$\min_{x \in X} \max_{y \in Y} F(x,y) \tag{3.39}$$

değerini aramaktadır.

Diğer taraftan D rakibinden alacağı paranın miktarını maksimize etmek için çalışmaktadır. Eğer $y \in Y$ stratejisini seçerse, en kötü durumda (P çok zeki veya çok şanslı) P'den alacağı miktar

$$F_*(y) = \min_{x \in X} F(x,y) \tag{3.40}$$

olacaktır. D'nin optimal stratejisi en kötü durumda alacağı parayı maksimize etmek olduğundan,

$$\max_{y \in Y} F_*(y) \tag{3.41}$$

problemini çözmek zorundadır. Bu problem de bir min-max problemidir çünkü,

$$\max_{y \in Y} \min_{x \in X} F(x, y) \tag{3.42}$$

değerini aramaktadır.

Her iki problem de birbirinin dualidir. P tarafından çözülecek olan min-max problemine *primal problem*, bu problemde minimize edilecek olan amaç fonksiyonu $F^*(x)$'ye *primal fonksiyon* adı verilir. Benzer şekilde, D tarafından çözülecek olan max-min problemine *dual problem*, bu problemde maksimize edilecek olan amaç fonksiyonu $F_*(x)$'ye *dual fonksiyon* adı verilmektedir.

Herhangi $x \in X$ ve $y \in Y$ için

$$F_*(y) = \min_{x \in X} F(x, y) \leq F(x, y) \leq \max_{y \in Y} F(x, y) = F^*(x) \tag{3.43}$$

yazılabilir ki buna *Zayıf Dualite Teoremi* denmektedir, yani kısaca $F_*(y) \leq F^*(x)$. Zayıf dualitenin bir sonucu olarak max-min probleminin optimal çözümünün, min-max probleminin optimal değeri tarafından üstten sınırlandırıldığı söylenebilir. Başka bir deyişle

$$\max_{y \in Y} \min_{x \in X} F(x, y) \leq \min_{x \in X} \max_{y \in Y} F(x, y) \tag{3.44}$$

Kuvvetli Dualite Problemi: Kuvvetli dualite şartı olan

$$\min_{x \in X} \max_{y \in Y} F(x, y) = \max_{y \in Y} \min_{x \in X} F(x, y) \tag{3.45}$$

şartını sağlaması ancak ve ancak F için semer noktası şartını sağlayan (x^*, y^*) çiftinin bulunmasıyla mümkündür. Diğer taraftan, $min - max = max - min$ eşitliği her zaman gerçekleşmeyebilir.

3.3.2 Lagrange Dualliği

Min-max dualliği doğrusal olmayan programlama için dual problemin geliştirilmesinde bir temel teşkil etmektedir. Anafikir şu şekildedir: F gibi bir ödeme fonksiyonu olan öyle bir oyun tanımlanmalıdır ki böylece F fonksiyonuna göre min-max probleminin çözümü aynı zamanda doğrusal olmayan minimizasyon probleminin çözümü olmalıdır. Sonuçta elde edilen max-min problemi de bunun duali olur. Bunu gerçekleştirmek için çeşitli yaklaşımlar vardır. Burada Lagrange yaklaşımı ele alınacaktır. Aşağıdaki gibi doğrusal olmayan problemi ele alalım:

$$\begin{gathered} \min_{\mathbf{x}} f(\mathbf{x}) \\ Kısıtlar: \mathbf{g}(\mathbf{x}) \geq \mathbf{0} \end{gathered} \tag{3.46}$$

Burada $\mathbf{x} \in X \in \mathbb{R}^n$ olup $\mathbf{g}(\mathbf{x})$ vektörü de problemdeki m adet eşitsizlik kısıtlarına ilişkin fonksiyonların oluşturduğu $m \times 1$ boyutlu bir vektördür. Lagrange ifadesi yazılırsa

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) - \boldsymbol{\lambda}^T g(\mathbf{x}) \tag{3.47}$$

Burada $\boldsymbol{\lambda} \in \mathbb{R}^m$ ve $\boldsymbol{\lambda} \geq \mathbf{0}$. Birazdan görüleceği gibi bu şekildeki bir doğrusal olmayan optimizasyon problemi bir min-max problemine dönüştürülebilir. Burada Lagrange fonksiyonu olan $\mathcal{L}(\mathbf{x}, \boldsymbol{\lambda})$ min-max problemindeki $F(x, y)$'nin $\boldsymbol{\lambda} \in \mathbb{R}^m$, $\boldsymbol{\lambda} \geq \mathbf{0}$ şeklindeki $\boldsymbol{\lambda}$ da y'nin rolünü üstlenecektir.

Öncellikle primal amaç fonksiyonunu tanımlayalım:

$$\mathcal{L}^*(\mathbf{x}) = \max_{\boldsymbol{\lambda} \geq \mathbf{0}} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) \tag{3.48}$$

Bu fonksiyona, sabit $\mathbf{x}$ için daha yakından bakarsak:

$$\mathcal{L}^*(\mathbf{x}) = \max_{\boldsymbol{\lambda}\geq\mathbf{0}}[f(\mathbf{x}) - \boldsymbol{\lambda}^T\mathbf{g}(\mathbf{x})] \tag{3.49}$$

yazılabilir. Eğer $\mathbf{g}(\mathbf{x}) \geq \mathbf{0}$ ise $\boldsymbol{\lambda}^T\mathbf{g}(\mathbf{x})$ ifadesi negatif olmayacaktır. Bu durumda $\mathcal{L}^*(\mathbf{x})$, $\boldsymbol{\lambda}^T\mathbf{g}(\mathbf{x})$ ifadesi sıfır olduğundan maksimum olacaktır. Bu örneğin $\boldsymbol{\lambda} = \mathbf{0}$ ile mümkün olabilir ki bu durumda da $\mathcal{L}^*(\mathbf{x}) = f(\mathbf{x})$ olur. Diğer taraftan, herhangi bir kısıt için $g_i(\mathbf{x}) < 0$ ise, bu kısıta ilişkin Lagrange çarpanı olan λ_i istenildiği kadar arttırılarak (diğer çarpanlar sıfır iken) Lagrange ifadesinin sınırsız bir şekilde artması sağlanabilir. Bu yüzden

$$\mathcal{L}^*(\mathbf{x}) = \begin{cases} f(\mathbf{x}), & \mathbf{g}(\mathbf{x}) \geq 0 \\ \infty, & diğer \end{cases} \tag{3.50}$$

yazılabilir. Bu durumda min-max problemi,

$$\min_{\mathbf{x}\in X} \mathcal{L}^*(\mathbf{x}) \tag{3.51}$$

şeklinde yazılabilir. $\mathcal{L}^*(\mathbf{x})$ 'ın sonsuz olduğu bölgeler ihmal edilirse, bu problem orjinal kısıtlı problemimiz haline gelir:

$$\begin{gathered} \min_{\mathbf{x}} f(\mathbf{x}) \\ Kısıtlar: \mathbf{g}(\mathbf{x}) \geq \mathbf{0} \end{gathered} \tag{3.52}$$

Böylece orjinal kısıtlı doğrusal olmayan problem bir min-max problemi şeklinde yazılmış olur. Artık min-max dualitesi kullanılarak problem yazılabilir. Herhangi bir $\boldsymbol{\lambda} \geq \mathbf{0}$ için dual fonksiyon şu şekilde yazılabilir:

$$\mathcal{L}_*(\boldsymbol{\lambda}) = \min_{\mathbf{x}\in X} \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}) \tag{3.53}$$

Sonuçta elde edilen max-min dual problemi şu şekilde olacaktır:

$$\mathcal{L}_*(\boldsymbol{\lambda}) = \max_{\boldsymbol{\lambda}\geq\mathbf{0}} \min_{\mathbf{x}\in X}[f(\mathbf{x}) - \boldsymbol{\lambda}^T\mathbf{g}(\mathbf{x})] \tag{3.54}$$

Bu bölüm boyunca kullanılacak olan kısıtların eşitsizlik kısıtları olduğu varsayılmaktadır. Eşitlik kısıtları için bir zorluk yoktu. Eğer kısıtlardan herhangi birinin mutlaka sıfıra eşit olması gerekiyorsa o kısıta ilişkin λ_i üzerindeki kısıt ($\lambda_i \geq 0$) kaldırılır.

Zayıf Duallik: Aşağıdaki gibi bir primal problemi ele alalım.

$$\begin{gathered}\min_{\mathbf{x}} f(\mathbf{x}) \\ Kısıtlar: \mathbf{g}(\mathbf{x}) \geq \mathbf{0}\end{gathered} \tag{3.55}$$

Bu problemin çözümü $\mathbf{x}$ olsun. $(\bar{\mathbf{x}}, \bar{\boldsymbol{\lambda}})$ çifti de bu probleme ilişkin dual problemin bir çözümü olsun. Bu durumda, $f(\bar{\mathbf{x}}) - \bar{\boldsymbol{\lambda}}^T\mathbf{g}(\bar{\mathbf{x}}) \leq f(\mathbf{x})$ yazılabilir.

Zayıf Duallik Teoreminin bir sonucu olarak,

$$\max_{\boldsymbol{\lambda}\geq\mathbf{0}} \mathcal{L}_*(\boldsymbol{\lambda}) \leq \min_{\mathbf{x}\in X}\{f(\mathbf{x}): \mathbf{g}(\mathbf{x}) \geq \mathbf{0}\} \tag{3.56}$$

yazılabilir ki bunun anlamı şudur: primal probleminin amaç fonksiyonunun optimum değeri dual problemin optimum değerine eşit veya ondan büyüktür. Kuvvetli Duallik Teoremine göre; primal ve dual problemlerin optimum değerleri, $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ gibi bir noktanın tüm $\mathbf{x} \in X$ ve $\boldsymbol{\lambda} \geq \mathbf{0}$ değerleri için $\mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}) \leq \mathcal{L}(\mathbf{x}^*, \boldsymbol{\lambda}^*) \leq \mathcal{L}(\mathbf{x}, \boldsymbol{\lambda}^*)$ şeklindeki semer noktası şartını sağlamasıyla birbirine eşit olur.

Semer noktası şartını sağlaması garanti olan bir tip problem vardır ve biçimi eşitlik (3.57)'de verilmiştir.

$$\begin{gathered}\min_{\mathbf{x}} f(\mathbf{x}) \\ Kısıtlar: g_i(\mathbf{x}) \geq \mathbf{0}, i = 1, \dots, m\end{gathered} \tag{3.57}$$

Burada $f(\mathbf{x})$ konveks bir fonksiyondur ve her bir aaa konkav bir fonksiyondur. Bu tüm problemlere *konveks programlama problemi* adı verilmektedir.

3.3.2.1 Karesel Programlama (Quadratic Programming-QP)

Karesel programlama problemi, amaç fonksiyonunun tasarım değişkenlerine göre karesel (quadratic), kısıtların ise doğrusal olduğu özel bir durumdur. Bir QP probleminin genel biçimi denklem (3.58)'de gösterilmiştir.

$$\begin{gathered} \min_{\mathbf{x}} f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{G}\mathbf{x} \\ Kısıtlar: \mathbf{A}\mathbf{x} \geq \mathbf{B} \\ \mathbf{C}\mathbf{x} = \mathbf{D} \end{gathered} \tag{3.58}$$

Burada $\mathbf{x} \in \mathbb{R}^n$ tasarım değişkenleri vektörü olup **H**,**G**,**A** ve **B** matrisleri uygun boyutlardadır. Tasarım değişkenlerine göre karesel bir fonksiyonu her zaman $\frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x} + \mathbf{G}\mathbf{x}$ biçimine getirmek mümkündür.

Karesel fonksiyonun bir özelliği de gradyan vektörünün ve Hessian matrisinin denklem (3.59)'daki gibi olmasıdır.

$$\nabla f(\mathbf{x}) = \mathbf{H}\mathbf{x} + \mathbf{G} \;\; ve \; \nabla^2 f(\mathbf{x}) = \mathbf{H} \tag{3.59}$$

4. DESTEK VEKTÖR MAKİNELERİ

Destek vektör makineleri, istatiksel öğrenme teorisi ve yapısal riski en aza indirme ilkesine dayanan, sınıflandırma ve bağlanım problemlerinin çözümü amacıyla Vapnik tarafından ortaya atılmış bir öğrenme yöntemidir. Destek vektör makineleri herhangi bir sınıflandırma ya da bağlanım problemini, bir karesel programlama problemine dönüştürerek yerel minimumlara takılmadan çözerler.

4.1 Destek Vektör Makineleriyle Sınıflandırma

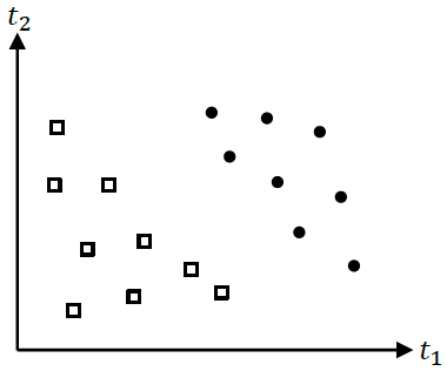
Destek vektörü sınıflandırmasının amacı yüksek boyutlu uzaylarda iyi öğrenen ayırıcı düzlemlerin verimli ve hesaplanabilir bir biçimde tasarlanmasıdır. Bu bölümde sınıflandırma problemlerine çözüm getirmek üzere tasarlanmış ve birçok farklı alanda uygulamaları bulunan destek vektör makineleri açıklanacaktır (Cristianini ve Taylor 2000, Vapnik 1995, Vapnik 1998[a]).

4.1.1 Doğrusal Sınıflandırma

Verilerin doğrusal olarak ayrıldığı durumda doğrusal sınıflandırıcılar kullanılmaktadır. Doğrusal sınıflandırıcı yöntemleri alt bölümlerde anlatılmaktadır.

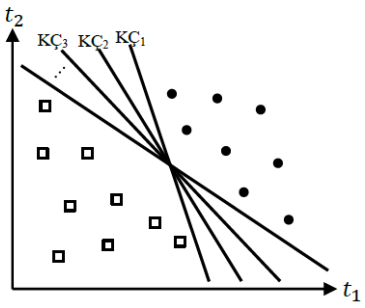
4.1.1.1 En Büyük Marjinli Sınıflandırıcı

$\mathbf{t}_i \in \mathbb{R}^n$ ve $y_i \in \{-1, +1\}$ olmak üzere $\mathcal{D}_{2-\text{Doğrusal}} = \{\mathbf{t}_i, y_i\}_{i=1}^{N}$ şeklinde verilen bir iki sınıflı doğrusal sınıflandırma verisini ele alalım. Burada problem, bu iki sınıflı veriyi uygun bir çizgi ile birbirinden ayırmaktır. Kolaylık olması açısından giriş uzayının boyutu iki olarak alınırsa, giriş verileri Şekil 4-1'e benzer bir şekilde olacaktır (İplikçi 2013).



Şekil 4-1: İki sınıflı doğrusal sınıflandırma verisi

Burada içi dolu yuvarlaklar $+1$ sınıfı verileri, içi boş kareler -1 sınıfı verileri temsil etsin. Şekil 4-2'den de görüleceği gibi bu iki sınıfı birbirinden ayıran çok sayıda "Karar Çizgisi (KÇ)" mevcuttur (İplikçi 2013).



Şekil 4-2: Mevcut karar çizgileri

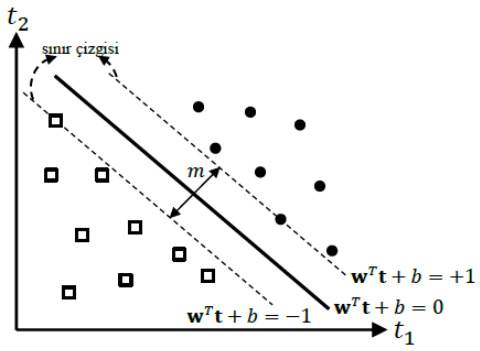
Her bir karar çizgisi denklem (4.1)'deki gibi bir doğru denklemi ile ifade edilebilir.

$$w_1 t_1 + w_2 t_2 + b = 0 \tag{4.1}$$

Burada w_i'ler ve b terimleri karar çizgisini belirleyen katsayılardır. $\mathbf{w}^T = [w_1\ w_2]$ olmak üzere, bir karar çizgisi şu şekilde de ifade edilebilir.

$$\mathbf{w}^T\mathbf{t} + b = \langle \mathbf{w}, \mathbf{t} \rangle + b = 0 \tag{4.2}$$

Burada, $\langle , \rangle$ operatörü iç çarpımı göstermektedir. Bu sınıflandırma problemini çözen çok sayıda karar çizgilerinden hangisinin seçilmesi uygundur? Sınıflandırma probleminde verilen verilerin gürültü olduğu yani ölçüm hataları gibi verilere bir takım bozucu etkilerin karışması ihtimali göz önüne alındığında en uygun karar çizgisinin sınıflara olabildiğince uzakta olması gerektiği görülür. Öyle bir karar çizgisi bulunmalıdır ki, bu çizgiye paralel olan sınır çizgileri arasındaki uzaklık *m*, buna *geometrik marjin* denmektedir, en büyük olsun. Şekil 4-3'te geometrik marjin görülmektedir (İplikçi 2013).



Şekil 4-3: Geometrik marjin

Böylece mevcut karar çizgileri arasından en güvenli olanı seçilmiş olur ki bu karar çizgisine *En Büyük Marjinli Sınıflandırıcı (Maximum Margin Classifier - MMC)* adı verilmektedir. Bu karar çizgisini bulmak için önce marjin büyüklüğünü hesaplayalım. Pozitif taraftaki herhangi bir $\mathbf{t}^+$ noktasının $\mathbf{w}^T\mathbf{t} + b = 0$ şeklindeki bir karar çizgisine olan Öklit uzaklığı $\frac{\mathbf{w}^T\mathbf{t}^+ + b}{\|\mathbf{w}\|}$ şeklindedir, burada $\|.\|$ operatörü $\|\mathbf{w}\| = \sqrt{\mathbf{w}^T\mathbf{w}}$ şeklinde verilen Öklit normudur. Bir $(\mathbf{w}, b)$ çifti ile verilen bir karar çizgisi $\alpha \in \mathbb{R}$ olmak üzere $(\alpha\mathbf{w}, \alpha b)$ şeklinde ölçeklenirse bu karar çizgisi değişmez ama $\mathbf{t}^+$ noktasının $\alpha\mathbf{w}^T\mathbf{t} + \alpha b = 0$ şeklindeki karar çizgisine olan Öklit uzaklığı değişir. Aynı durum negatif taraftaki herhangi bir $\mathbf{t}^-$ noktası için de geçerlidir. Varsayalım ki, pozitif taraftaki sınır çizgisi üzerindeki bir noktanın $\mathbf{w}^T\mathbf{t} + b = 0$ şeklindeki karar çizgisine olan Öklit uzaklığı $+1$ ve negatif taraftaki sınır çizgisi üzerindeki bir noktanın aynı karar çizgisine olan Öklit uzaklığı -1 olacak şekilde bir ölçeklenme yapılmış olsun.

$$\mathbf{w}^T\mathbf{t}^+ + b = +1$$
$$\mathbf{w}^T\mathbf{t}^- + b = -1 \tag{4.3}$$

Böylece geometrik marjin şu şekilde bulunur.

$$m = \left(\frac{\mathbf{w}^T\mathbf{t}^+ + b}{\|\mathbf{w}\|}\right) - \left(\frac{\mathbf{w}^T\mathbf{t}^- + b}{\|\mathbf{w}\|}\right)$$
$$= \frac{1}{\|\mathbf{w}\|}[(\mathbf{w}^T\mathbf{t}^+ + b) - (\mathbf{w}^T\mathbf{t}^- + b)]$$
$$= \frac{1}{\|\mathbf{w}\|}(1 - b - (-1 - b)) \tag{4.4}$$
$$= \frac{2}{\|\mathbf{w}\|}$$

İki sınıflı sınıflandırma problemini çözen ve en büyük geometrik marjine sahip sınıflandırıcının bulunması problemi şu şekilde ifade edilebilir: $\mathbf{t}_i \in \mathbb{R}^2$ ve $y_i \in \{-1, +1\}$ olmak üzere $\mathcal{D}_{2-\text{Doğrusal}} = \{\mathbf{t}_i, y_i\}_{i=1}^N$ şeklinde verilen bir iki sınıflı doğrusal sınıflandırma verisi için, $\mathbf{w}^T\mathbf{t} + b = 0$ şeklinde öyle bir sınıflandırıcı bulunmalıdır ki sınıflandırıcı verileri hatasız şekilde ayırmalı yani, $y_i = +1$ iken $\mathbf{w}^T\mathbf{t}_i + b \geq +1$ olmalı, $y_i = -1$ iken $\mathbf{w}^T\mathbf{t}_i + b \leq -1$ olmalı ve aynı zamanda $\frac{1}{\|\mathbf{w}\|}$ geometrik marjini en büyük olmalıdır. Bu problem eşitsizlik-kısıtlı bir problem olarak (4.5) eşitliğindeki gibi ifade edilebilir:

$$\max_{\mathbf{w},b} \frac{2}{\|\mathbf{w}\|}$$
$$Kıs: \mathbf{w}^T\mathbf{t}_i + b \geq +1\,, \qquad y_i = +1\ ise \tag{4.5}$$
$$\mathbf{w}^T\mathbf{t}_i + b \leq -1, \qquad y_i = -1\ ise$$

Aslında $\frac{1}{\|\mathbf{w}\|}$ fonksiyonunu en büyük yapmak için $\|\mathbf{w}\|$ fonksiyonunu ya da kolaylık olması açısından $\frac{1}{2}\|\mathbf{w}\|^2$ fonksiyonunu en küçük yapmak yeterlidir. Bu durum dikkate alınır ve $y_i = +1$ ve $y_i = -1$ için ayrı ayrı yazılan kısıtlar, $y_i(\mathbf{w}^T\mathbf{t}_i + b) \geq +1$ şeklinde birleştirilirse, eşitsizlik-kısıtlı bir problem şu hale gelir:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2$$
$$Kıs: y_i \mathbf{w}^T \mathbf{t}_i + b \geq +1\,, \qquad i = 1, \ldots, N \tag{4.6}$$

Bu şekilde verilen problem esasında bir Karesel Programlama problemidir. Şimdi bu primal formülasyonun dualini elde etmek için Lagrange ifadesini yazalım:

$$\begin{aligned} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\lambda}) &= f(\mathbf{x}) - \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x}) \\ &= \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \lambda_i [y_i(\mathbf{w}^T \mathbf{t}_i + b) - 1] \\ &= \frac{1}{2}\mathbf{w}^T\mathbf{w} - \sum_{i=1}^{N} \lambda_i y_i (\mathbf{w}^T \mathbf{t}_i + b) + \sum_{i=1}^{N} \lambda_i \\ &= \frac{1}{2}\mathbf{w}^T\mathbf{w} - \mathbf{w}^T \sum_{i=1}^{N} \lambda_i y_i \mathbf{t}_i - b \sum_{i=1}^{N} \lambda_i y_i + \sum_{i=1}^{N} \lambda_i \end{aligned} \tag{4.7}$$

Bu Lagrange ifadesinin birincil değişkenlere $(\mathbf{w}, b)$ göre türevi alınıp, sıfıra eşitlenirse aşağıdaki eşitlikler elde edilir.

$$\begin{aligned} \frac{\partial \mathcal{L}(\mathbf{w}, b, \boldsymbol{\lambda})}{\partial \mathbf{w}} = 0 \quad \mathbf{w} &= \sum_{i=1}^{N} \lambda_i y_i \mathbf{t}_i \\ \frac{\partial \mathcal{L}(\mathbf{w}, b, \boldsymbol{\lambda})}{\partial b} = 0 \quad \sum_{i=1}^{N} \lambda_i y_i &= 0 \end{aligned} \tag{4.8}$$

Burada Lagrange çarpanlarının $\boldsymbol{\lambda} \geq \mathbf{0}$ şartlarını sağlamaları gerektiği unutulmamalıdır. Denklem (4.9)'da bulunan eşitlikler Lagrange ifadesinde yerine konulduğunda,

$$\begin{aligned} \mathcal{L}(\mathbf{w}) &= \frac{1}{2}\mathbf{w}^T\mathbf{w} - \mathbf{w}^T \sum_{i=1}^{N} \lambda_i y_i \mathbf{t}_i - b \sum_{i=1}^{N} \lambda_i y_i + \sum_{i=1}^{N} \lambda_i \\ &= -\frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{N} \lambda_i \end{aligned} \tag{4.9}$$

$$= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y_i y_j \mathbf{t}_i^T\mathbf{t}_j + \sum_{i=1}^{N}\lambda_i$$

Artık iki sınıflı sınıflandırma probleminin dual formülasyonunu şu şekilde yazabiliriz.

$$\min_{\boldsymbol{\lambda}} \mathcal{L}(\boldsymbol{\lambda}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\lambda_i\lambda_j y_i y_j \mathbf{t}_i^T\mathbf{t}_j - \sum_{i=1}^{N}\lambda_i$$

$$Kıs: \sum_{i=1}^{N}\lambda_i y_i = 0 \tag{4.10}$$

$$\boldsymbol{\lambda} \geq \mathbf{0}$$

Bu formülasyon da bir karesel programlama problemi şeklindedir. Aynı problemin karmaşıklığı, primal formülasyonla ifade edilebildiğinde giriş uzayının boyutu (n) ile orantılı iken dual formülasyonla ifade edildiğinde veri sayısı (N) ile orantılı olmaktadır. Bu problemin çözümünden N adet negatif olmayan Lagrange çarpanı (λ_i) elde edilir ama bunlardan bazıları sıfırdır. $\lambda_i > 0$ şeklinde sıfır Lagrange çarpanına karşı düşen $\mathbf{t}_i$ vektörüne *destek vektörü* denmektedir. Destek vektörleri $y_i(\mathbf{w}^T\mathbf{t}_i + b) = 1$ şartını sağlarlar ve sınır çizgilerinde yer alırlar. Primal uzayda $\hat{y}(\mathbf{t}) = \mathbf{w}^T\mathbf{t} + b$ şeklinde olan sınıflandırıcı modeli, $\mathbf{w} = \sum_{i=1}^{N}\lambda_i y_i \mathbf{t}_i$eşitliği ile dual uzaydaki ifadesi şu hale gelir (Vapnik 1998[a]):

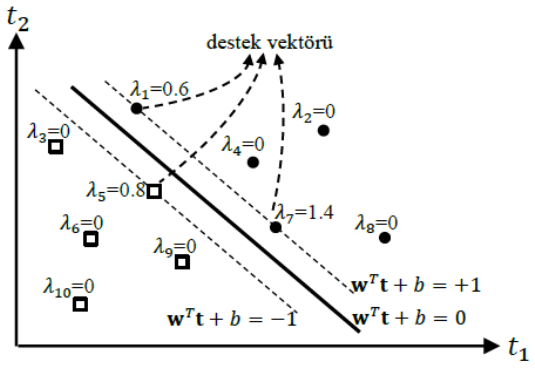
$$\hat{y}(\mathbf{t}) = \sum_{j=1}^{N}\lambda_i\lambda_j y_i y_j \mathbf{t}_i^T\mathbf{t}_j + b \tag{4.11}$$

Buradaki b terimi, tüm destek vektörlerinin $y_i(\mathbf{w}^T\mathbf{t}_i + b) = 1$ şartını sağlaması gerekliliği kullanılarak bulunur. Önceden de belirtildiği gibi karesel programlama probleminin çözümünden elde edilen N adet Lagrange çarpanlarından bazıları sıfırdır ve yukarıdaki sınıflandırıcı modelinde çıkışa bir katkısı yoktur. Dolayısıyla, sınıflandırıcı modelinde sadece destek vektörleri dikkate alınarak $\hat{y}(\mathbf{t}) = \sum_{i\in\mathcal{S}}\lambda_i y_i \mathbf{t}_i^T\mathbf{t} + b$ şeklinde daha seyrek bir model elde edilir, burada $\mathcal{S}$ kümesi sadece destek vektörlerinin indekslerinden oluşan bir kümedir ve bu modele *destek*

vektör modeli (support vector model-SVM model) denmektedir. $\mathbf{w} = \sum_{i=1}^{N} \lambda_i y_i \mathbf{t}_i$ eşitliğinden görüldüğü gibi sınıflandırıcının katsayıları verilerin doğrusal bir kombinasyonudur. SVM modeli daha az sayıda veri içermesi bakımından seyrek bir modeldir çünkü karar çizgisi sadece destek vektörleri tarafından belirlenmektedir. Geri kalan verilerin karar çizgisinde bir etkisi yoktur. **z** gibi bir test girişine karşı SVM modelinin çıkışı denklem (4.12) ile hesaplanır.

$$\hat{y}(\mathbf{z}) = \sum_{i \in S} \lambda_i y_i \mathbf{t}_i^T \mathbf{z} + b \tag{4.12}$$

Eğer çıkış$+1$'den büyükse +1 olarak, -1'den küçükse de -1 olarak kabul edilir. Bu doğrusal SVM sınıflandırıcısının geometrik yorumu aşağıdaki şekilde daha rahat görülebilir (İplikçi 2013).



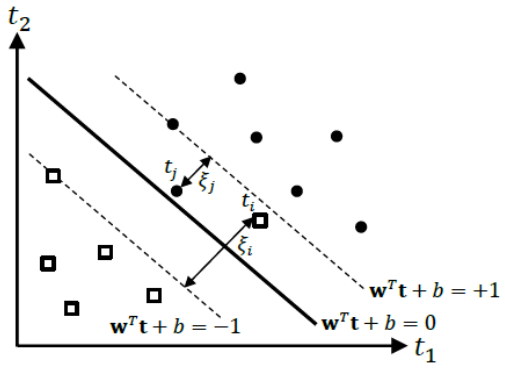
Şekil 4-4: Doğrusal SVM sınıflandırıcısının geometrik yorumu

Şekil 4-4'ten de görüldüğü gibi destek vektörleri sınır çizgileri üzerindedir ve onlara ilişkin Lagrange katsayıları sıfırdan büyüktür. Destek vektörleri dışındaki diğer verilere ilişkin Lagrange katsayıları sıfırdır. Dolayısıyla SVM modelini sadece destek vektörleriyle ifade etmek mümkündür.

4.1.1.2 Esnek Marjinli Sınıflandırıcı

Görüldüğü gibi iki sınıflı doğrusal bir sınıflandırma problemi bir doğrusal SVM modeli ile kolaylıkla çözülebilmektedir. Ancak, Şekil 4.5'te görüldüğü gibi, doğrusal olarak ayrılabilen verilerin bazılarının karşı tarafa geçtiği durumlar olabilir.

Bu durumda bazı verilerin sınır çizgileri arasında kalması hatta yanlış sınıflandırılması pahasına da olsa hala doğrusal SVM sınıflandırıcısı kullanılabilir.



Şekil 4-5: Esnek marjinli sınıflandırıcının kullanıldığı durum

Şekil 4-5'ten de görüldüğü gibi, sınır çizgileri arasında kalan verilerin bazıları (t_j gibi) doğru sınıflandırılırken, bazıları da (t_i gibi) yanlış sınıflandırılabilmektedir (İplikçi 2013). Sınır çizgileri arasında kalan verileri amaç fonksiyonu içerisinde cezalandırmak yoluyla problemi deklem (4.13) şeklinde ifade edebiliriz (Vapnik 1995, Vapnik 1998[a]).

$$\min_{\mathbf{w},b} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i$$

$$Kıs: y_i(\mathbf{w}^T\mathbf{t}_i + b) \geq 1 - \xi_i, \quad i = 1, \dots, N \tag{4.13}$$

$$\xi_i \geq 0, \quad i = 1, \dots, N$$

Burada negatif olmayan ξ_i büyüklüğü $\mathbf{t}_i$ verisi için sınıflandırıcının yaptığı hata miktarıdır ve *gevşek değişken* olarak adlandırılmaktadır. *C* büyüklüğü ise hatları cezalandıran bir ceza parametresidir. Görüldüğü gibi, herhangi bir $\mathbf{t}_i$ verisi için sınıflandırıcının ξ_i kadar hata yapmasına izin verilmektedir. Ancak bu hata amaç fonksiyonunda cezalandırılmaktadır. Bu yüzden sınıflandırıcıya *Esnek Marjinli Sınıflandırıcı* adı verilmektedir. Primal formda bu şekilde verilen sınıflandırma problemini şimdi dual forma dönüştürmek için Lagrange ifadesini yazalım:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i[y_i(\mathbf{w}^T\mathbf{t}_i + b) - 1 + \xi_i] - \sum_{i=1}^{N}\beta_i\xi_i$$

$$= \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i y_i(\mathbf{w}^T\mathbf{t}_i + b) + \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}\alpha_i\xi_i - \sum_{i=1}^{N}\beta_i\xi_i \tag{4.14}$$

$$= \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i - \mathbf{w}^T\sum_{i=1}^{N}\alpha_i y_i\mathbf{t}_i - b\sum_{i=1}^{N}\alpha_i\, y_i + \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}(\alpha_i + \beta_i)\xi_i$$

Burada Lagrange çarpanları α_i ve β_i'lerdir. Bu Lagrange ifadesinin birincil değişkenlere ($\mathbf{w}$,b,$\boldsymbol{\xi}$) göre türevi alınıp sıfıra eşitlenirse aşağıdaki eşitlikler elde edilir.

$$\frac{\partial\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial\mathbf{w}} = 0 \quad \mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i\mathbf{t}_i$$

$$\frac{\partial\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial b} = 0 \quad \sum_{i=1}^{N}\alpha_i y_i = 0 \tag{4.15}$$

$$\frac{\partial\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}, \boldsymbol{\beta})}{\partial\xi_i} = 0 \quad C - (\alpha_i + \beta_i) = 0, \;\; i = 1, \ldots, N$$

Burada Lagrange çarpanlarının $\boldsymbol{\alpha}, \boldsymbol{\beta} \geq \mathbf{0}$ şartlarını sağlamaları gerektiği unutulmamalıdır. Yukarıda bulunan eşitlikler Lagrange ifadesinde yerine konulduğunda denklem (4.16) elde edilir.

$$\mathcal{L}(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}\xi_i - \mathbf{w}^T\sum_{i=1}^{N}\alpha_i y_i\mathbf{t}_i - b\sum_{i=1}^{N}\alpha_i\, y_i + \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}(\alpha_i + \beta_i)\xi_i$$

$$= -\frac{1}{2}\mathbf{w}^T\mathbf{w} + \sum_{i=1}^{N}\alpha_i \tag{4.16}$$

$$= -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j\mathbf{t}_i^T\mathbf{t}_j - \sum_{i=1}^{N}\alpha_i$$

Bu lagrange ifadesinin son satırından da görüldüğü gibi, yerine koymalar sonucunda **β** Lagrange çarpanı ortadan kalkmıştır. Böylece, iki sınıflı sınıflandırma probleminin dual formülasyonu denklem (4.17)'deki gibi yazılabilir.

$$\min_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j \mathbf{t}_i^T \mathbf{t}_j - \sum_{i=1}^{N} \alpha_i$$

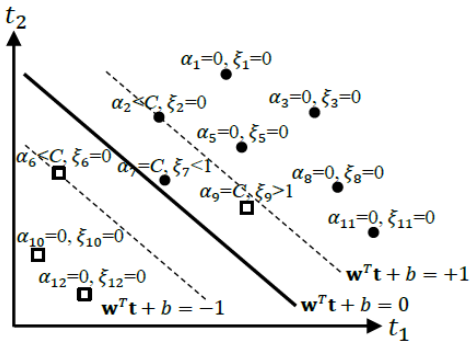
$$Kıs: \sum_{i=1}^{N} \alpha_i y_i = 0 \tag{4.17}$$

$$0 \leq \alpha_i \leq C, \;\; i = 1, \ldots, N$$

Problem bu haliyle En Büyük Marjinli Sınıflandırıcı problemine çok benzemektedir. Bu da bir karesel programlama problemidir. En Büyük Marjinli Sınıflandırıcı için söylenen tüm ifadeler Esnek Marjinli Sınıflandırıcı için de geçerlidir. Benzer şekilde, Esnek Marjinli Sınıflandırıcının dual uzaydaki ifadesi denklem (4.18)'deki gibidir.

$$\hat{y}(\mathbf{t}) = \sum_{i \in \mathcal{S}} \alpha_i y_i \mathbf{t}_i^T \mathbf{t} + b \tag{4.18}$$

Aradaki tek fark, Esnek Marjinli Sınıflandırıcı'da Lagrange çarpanları üzerinde $0 \leq \alpha_i \leq C$ şeklinde bir üst sınır olmasıdır. Bu sınıflandırıcının geometrik yorumu Şekil 4-6'da görülmektedir (İplikçi 2013).

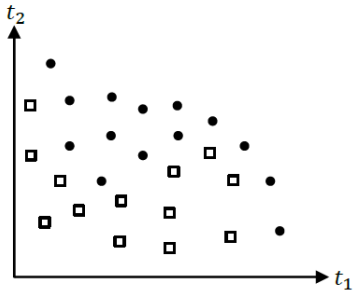


Şekil 4-6: Esnek marjinli sınıflandırıcının geometrik yorumu

Şekil 4-6'da görüldüğü gibi, sınır çizgileri dışında kalan verilerin Lagrange çarpanları (α_i'ler) ve bu veriler için sınıflandırıcının yaptığı hatalar (ξ_i'ler) sıfırdır. Sınır çizgilerinde yer alan destek vektörlerinin Lagrange çarpanları ceza teriminden (C) küçükken bu veriler için sınıflandırıcının yaptığı hatalar sıfırdır. Sınır çizgileri içinde yer alan verilerin Lagrange çarpanları ceza terimine eşittir ancak doğru sınıflandırılan veri için sınıflandırıcının yaptığı hata birden küçükken doğru sınıflandırılmayan veri için sınıflandırıcının yaptığı hata birden büyük olmaktadır.

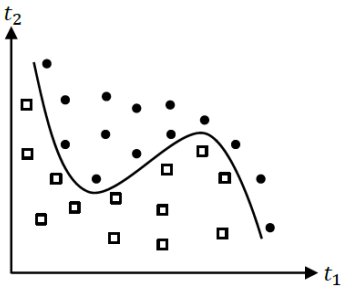
4.1.2 Doğrusal Olmayan Sınıflandırma

Verilerin doğrusal olarak ayrılamadığı pek çok durumda artık doğrusal sınıflandırıcılar işe yaramamaktadır. Bu durumda doğrusal olmayan sınıflandırıcıları kullanmak gerekir. Öncelikle doğrusal olmayan sınıflandırma problemini tanımlayalım: $\mathbf{t}_i \in \mathbb{R}^n$ ve $y_i \in \{-1, +1\}$ olmak üzere $\mathcal{D}_{2-\text{Doğrusal}-\text{olmayan}} = \{\mathbf{t}_i, y_i\}_{i=1}^{N}$ şeklinde verilen bir iki sınıflı doğrusal olmayan sınıflandırma verisini ele alalım. Burada problem, iki sınıflı veriyi uygun bir sınıflandırıcı ile birbirinden ayırmaktır. Kolaylık olması açısından giriş uzayının boyutu iki olarak alınırsa, giriş verileri Şekil 4-7'ye benzer şekilde olacaktır. Burada içi dolu yuvarlaklar $+1$ sınıfı verileri, içi boş kareler de -1 sınıfı verileri temsil etsin (İplikçi 2013).



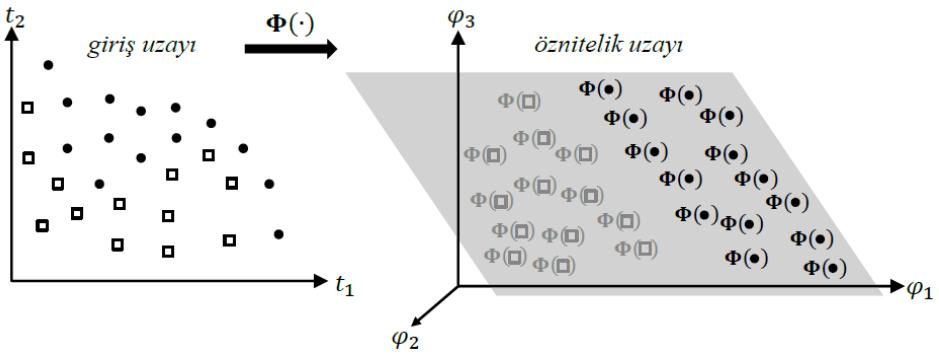
Şekil 4-7: İki sınıflı doğrusal olmayan sınıflandırma verisi

Şekil 4-8'den de görüleceği bu iki sınıfı birbirinden ayırmak için doğrusal olmayan bir "Karar Eğrisi (KE)" gerekmektedir ki bu eğriyi giriş uzayında elde etmek oldukça zor bir iştir.



Şekil 4-8: İki sınıfı birbirinden ayıran karar eğrisi

Doğrusal olmayan sınıflandırma problemini çözen uygun bir eğriyi giriş uzayı yerine daha yüksek boyutlu bir uzayda çözmek daha kolay olabilir. Başka bir deyişle, giriş uzayındaki her bir veri noktası, $\boldsymbol{\phi}(.)$ gibi uygun bir dönüşüm kullanılarak $\boldsymbol{\mathcal{F}}$ ile gösterilen öznitelik uzayı (feature space) olarak adlandırılacak daha yüksek boyutlu bir uzaya taşındığında daha kolay sınıflandırılabilir biçime hatta doğrusal olarak ayrılabilir hale gelebilir (Vapnik 1998[b]). Bu durum Şekil 4-9'da daha açık olarak görülebilir (İplikçi 2013).



Şekil 4-9: Giriş uzayından öznitelik uzayına geçiş

Öznitelik uzayında sınıflandırma işini yapan bu karar eğrisi, m öznitelik uzayının boyutu olmak üzere denklem (4.19) gibi bir doğrusal bir denklem ile ifade edilebilir.

$$\mathbf{w}^T\boldsymbol{\phi}(\mathbf{t}) = \langle \mathbf{w}, \boldsymbol{\phi}(\mathbf{t}) \rangle = 0 \tag{4.19}$$

Burada $\mathbf{w}^T = [w_1 \dots w_m]$ vektöründeki w_i 'ler karar eğrisini belirleyen katsayılardır, $\boldsymbol{\phi}(\mathbf{t})$ vektörü de öznitelik uzayında $\boldsymbol{\phi}(\mathbf{t}) = [\phi_1\ \phi_2\ \dots \phi_m]^T$ şeklinde m boyutlu bir vektördür. Dikkat edilirse burada bir b terimi kullanılmamıştır. Çünkü

bu terim zaten öznitelik içinde örneğin $\phi_1 = 1$ alınarak model içerisinde yer almaktadır.

Varsayalım ki giriş uzayında doğrusal olarak ayrılamayan veriler uygun bir $\boldsymbol{\phi}$**(.)** dönüşümüyle öznitelik uzayına taşındığında doğrusal olarak ayrılabilir hale gelmiş olsun. Bu durumda artık öznitelik uzayında doğrusal olan veriler, yine bu uzayda tanımlı doğrusal bir SVM sınıflandırıcısı ile ayrılabilirler. Giriş uzayındaki her bir $\mathbf{t}_i$ veri noktası artık öznitelik uzayında $\boldsymbol{\phi}(\mathbf{t}_i)$ şeklinde bir veri noktasına dönüşmüştür. Böylece bu sınıflandırma probleminin öznitelik uzayındaki dual formülasyonu doğrudan denklem (4.20)'deki gibi yazılır.

$$\min_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \boldsymbol{\phi}^T(\mathbf{t}_i)\boldsymbol{\phi}(\mathbf{t}_j) - \sum_{i=1}^{N}\alpha_i$$

$$Kıs: \sum_{i=1}^{N}\alpha_i y_i = 0 \tag{4.20}$$

$$0 \le \alpha_i \le C, \;\; i = 1, \dots, N$$

Problem bu haliyle doğrusal olarak ayrılabilen veriler için kullanılan Esnek Marjinli Sınıflandırıcı problemine çok benzemektedir. Öznitelik uzayında bu problemin çözümüyle elde edilecek olan SVM sınıflandırıcının modeli de denklem (4.21)'de görülmektedir.

$$\hat{y}(\mathbf{t}_j) = \sum_{i\in\mathcal{S}}\alpha_i y_i \boldsymbol{\phi}^T(\mathbf{t}_i)\boldsymbol{\phi}(\mathbf{t}_j) \tag{4.21}$$

Görüldüğü gibi, aradaki tek fark amaç fonksiyonundaki ve SVM modelindeki $\mathbf{t}_i^T\mathbf{t}_j$ iç çarpımı yerine öznitelik uzayındaki $\boldsymbol{\phi}^T(\mathbf{t}_i)\boldsymbol{\phi}(\mathbf{t}_j)$ iç çarpımı gelmiştir. Dolayısıyla, öznitelik uzayındaki $\boldsymbol{\phi}^T(\mathbf{t}_i)\boldsymbol{\phi}(\mathbf{t}_j)$ iç çarpımı bir şekilde bulunabilirse doğrusal olmayan sınıflandırma problemi de SVM yaklaşımı ile çözülebilir. Dikkat edilirse, bu çözüm için dönüşüm fonksiyonu olan $\boldsymbol{\phi}$**(.)**'nin kendisinin doğrudan bilinmesine gerek yoktur; onun yerine sadece $\boldsymbol{\phi}^T(\mathbf{t}_i)\boldsymbol{\phi}(\mathbf{t}_j)$ iç çarpımının bilinmesi yeterlidir. Bu noktada, $\boldsymbol{\phi}^T(\mathbf{t}_i)\boldsymbol{\phi}(\mathbf{t}_j)$ iç çarpımının bulunması için *Kernel (Çekirdek) Fonksiyonu* yaklaşımı kullanılmaktadır. Kernel fonksiyonu öznitelik uzayında iç

çarpıma denk düşen bir fonksiyondur ve Q(.) notasyonu ile gösterilir. Başka bir deyişle, öznitelik uzayında $\boldsymbol{\phi}^T(\mathbf{t}_i)\boldsymbol{\phi}(\mathbf{t}_j)$ şeklindeki iç çarpım ifadesi kernel fonksiyonu kullanılarak denklem (4.22) gibi ifade edilebilir.

$$\boldsymbol{\phi}^T(\mathbf{t}_i)\boldsymbol{\phi}(\mathbf{t}_j) = Q(\mathbf{t}_i, \mathbf{t}_j) = Q_{ij} \tag{4.22}$$

Görüldüğü gibi öznitelik uzayındaki iç çarpım kernel fonksiyonu ile halledilmektedir ve dolayısıyla dönüşüm fonksiyonu olan $\boldsymbol{\phi}$**(.)**'nin kendisi yerine uygun bir kernel fonsiyonunun bulunması SVM sınıflandırıcı probleminin çözülmesi açısından yeterlidir. Kernel fonksiyonu ile $\boldsymbol{\phi}$**(.)** dönüşüm fonksiyonu arasındaki ilişkiyi ifade etmek için $Q(\mathbf{t}_i, \mathbf{t}_j) = (1 + \mathbf{t}_i^T\mathbf{t}_j)^2$ şeklinde tanımlanan ikinci dereceden polinom kernel fonksiyonuna karşı düşen $\boldsymbol{\phi}$**(.)** dönüşüm fonksiyonunu bulalım. Kolaylık olması açısından giriş uzayının boyutunun 2 olduğu durumu ele alalım, yani $t_i, t_j \in \mathbb{R}^2$. Bu durumda, kernel fonksiyonu açılırsa, denklem (4.23) elde edilir (Saunders ve diğ. 1998).

$$\begin{aligned} Q(\mathbf{t}_i, \mathbf{t}_j) &= \left(1 + \mathbf{t}_i^T\mathbf{t}_j\right)^2 \\ &= 1 + t_{i1}^2 t_{j1}^2 + \mathbf{2}t_{i1}t_{j1}t_{i2}t_{j2} + t_{i2}^2 t_{j2}^2 + \mathbf{2}t_{i1}t_{j1} + \mathbf{2}t_{i2}t_{j2} \\ &= \begin{bmatrix} 1 & t_{i1}^2 & \sqrt{2}t_{i1}t_{i2} & t_{i2}^2 & \sqrt{2}t_{i1} & \sqrt{2}t_{i2} \end{bmatrix} \begin{bmatrix} 1 \\ t_{i1}^2 \\ \sqrt{2}t_{i1}t_{i2} \\ t_{i2}^2 \\ \sqrt{2}t_{i1} \\ \sqrt{2}t_{i2} \end{bmatrix} \\ &= \boldsymbol{\phi}^T(\mathbf{t}_i)\boldsymbol{\phi}(\mathbf{t}_j) \end{aligned} \tag{4.23}$$

Burada dönüşüm fonksiyonu $\boldsymbol{\phi}(\mathbf{t}) = \begin{bmatrix} 1 & t_{i1}^2 & \sqrt{2}t_{i1}t_{i2} & t_{i2}^2 & \sqrt{2}t_{i1} & \sqrt{2}t_{i2} \end{bmatrix}^T$ şeklindedir. Buradan da görülüyor ki 2 boyutlu bir giriş uzayından $\boldsymbol{\phi}(\mathbf{t})$ dönüşümüyle 6 boyutlu bir öznitelik uzayına geçilebilmektedir.

Bir kernel fonksiyonunun $\mathbf{t}_i$ ve $\mathbf{t}_j$ gibi iki argümanı vardır ve kernel fonksiyonu sezgisel olarak bu iki argüman arasındaki benzerliği temsil eder.

Örneğin, yukarıdaki $Q(\mathbf{t}_i, \mathbf{t}_j) = (1 + \mathbf{t}_i^T \mathbf{t}_j)^2$ şeklindeki kernel fonksiyonunu ele alalım. $\mathbf{t}_i$ ve $\mathbf{t}_j$ vektörleri birbirine ne kadar çok benzerse fonksiyon o kadar büyük, ne kadar az benzerlerse o kadar küçük değerler alır. Limit durumunda vektörler birbirinin aynısı iken en büyük değerini, vektörler birbirine dik iken de en küçük değerini alır. Kernel yaklaşımının bir diğer özelliği de argümanlarının nümerik değerler almak zorunda olmamasıdır. Yani nümerik değerler içeren $\mathbf{t}_i$ ve $\mathbf{t}_j$ vektörleri yerine metin, dizi, dilsel etiket gibi başka başka biçimlerde giriş verisi kullanılabilir ki bu da SVM yaklaşımlarının çok farklı alanlarda uygulanmasına olanak sağlar.

Literatürde önerilmiş pek çok kernel fonksiyonu vardır. Bunlardan bazıları şu şekildedir:

- *d^{inci} dereceden polinom kernel fonksiyonu:* $Q(\mathbf{t}_i, \mathbf{t}_j) = (1 + \mathbf{t}_i^T \mathbf{t}_j)^d$ şeklinde olup kernel parametresi aynı zamanda da polinomun derecesi olan *d*'dir.
- *Gauss kernel fonksiyonu:* $Q(\mathbf{t}_i, \mathbf{t}_j) = e^{-\frac{\|t_i - t_j\|^2}{2\sigma^2}}$ şeklinde olup kernel parametresi σ aynı zamanda da *genişlik parametresi* olarak adlandırılmaktadır.
- *Sigmoid kernel fonksiyonu:* $Q(\mathbf{t}_i, \mathbf{t}_j) = tanh(k\mathbf{t}_i^T \mathbf{t}_j + \theta)^d$ şeklinde olup kernel parametreleri k, θ ve *d*'dir.

Giriş uzayından öznitelik uzayına dönüşümü sağlayan $\boldsymbol{\phi}(\mathbf{t})$ dönüşüm fonksiyonunun olabilmesi için kernel fonksiyonunun sağlanması gereken koşullara *Mercel Koşulları* adı verilmektedir. Bu koşullar aynı zamanda bir fonksiyonun kernel fonksiyonu olabilmesi için gerek ve yeter koşullardır. Mercer teoreminde göre pozitif yarı tanımlı her fonksiyon kernel fonksiyonudur. Başka bir deyişle, denklem (4.24) sağlayan bir Q(,) fonksiyonu bir kernel fonksiyonudur.

$$\forall \mathbf{t}_i, \mathbf{t}_j \in \mathbb{R}^n \; için \; 0 \leq Q(\mathbf{t}_i, \mathbf{t}_j) = \boldsymbol{\phi}^T(\mathbf{t}_i)\boldsymbol{\phi}(\mathbf{t}_j) \tag{4.24}$$

Sınıflandırma problemine geri dönülürse, öznitelik uzayındaki dual formülasyon denklem (4.25) gibi yazılabilir.

$$\min_{\boldsymbol{\alpha}} \mathcal{L}(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j Q(\mathbf{t},\mathbf{t}_j) - \sum_{i=1}^{N}\alpha_i$$

$$Kıs: \sum_{i=1}^{N}\alpha_i y_i = 0 \tag{4.25}$$

$$0 \le \alpha_i \le C, \;\; i = 1, \dots, N$$

Benzer şekilde SVM sınıflandırıcı modeli de denklem (4.26) gibidir.

$$\hat{y}(\mathbf{t}) = \sum_{i\in\mathcal{S}}\alpha_i y_i Q(\mathbf{t},\mathbf{t}_j) \tag{4.26}$$

4.2 Destek Vektör Makineleriyle Bağlanım

$\mathbf{t}_i \in \mathbb{R}^R$ ve $y_i \in \mathbb{R}$ olmak üzere $\mathcal{D}_{\text{MISO}} = \{\mathbf{t}_i, y_i\}_{i=1}^{N}$ şeklinde verilen birçok girişli tek çıkışlı (Multi Input Single Output) veriyi ele alalım. Bağlanım problemi olarak ele alınan problemde amaç bu verilerin giriş-çıkış ilişkisini uygun bir model ile temsil etmektedir. SVM yaklaşımı ile bu bağlanım problemi çözülebilir. Sınıflandırma problemine benzer şekilde, bağlanım problemlerinin çözümünde de öznitelik uzayında doğrusal olan bir modelden yararlanılmaktadır. Bu model denklem (4.27) ile verilmektedir.

$$\hat{y}(\mathbf{t}) = \mathbf{w}^T\boldsymbol{\phi}(\mathbf{t}) + b \tag{4.27}$$

Burada $\boldsymbol{\phi}(.)$ sınıflandırma probleminde olduğu gibi giriş uzayından öznitelik uzayına bir dönüşüm fonksiyonudur. Görüldüğü gibi bu model giriş uzayında doğrusal değildir ancak öznitelik uzayında doğrusaldır.

4.2.1 ε-Duyarsız Destek Vektör Makineleriyle Bağlanım

ε-SVR algoritması, bu bağlanım modeliyle modellenebilir. Bağlanımda, yaklaşım yanılgısı kullanılır. Uygulamada çeşitli kayıp işlevleri bulunmaktadır. Burada kullanılacak kayıp işlevi ε-toleranslı kayıp işlevidir.

$$Y_\varepsilon = \begin{cases} 0, & eğer\ |y(\mathbf{t}) - \hat{y}(\mathbf{t})| < \varepsilon \\ |y(\mathbf{t}) - \hat{y}(\mathbf{t})| - \varepsilon, & eğer\ |y(\mathbf{t}) - \hat{y}(\mathbf{t})| \geq \varepsilon \end{cases} \tag{4.28}$$

Destek vektör makineleri ile bağlanımın ana fikri şudur: Kestirim işlevi $\hat{y}(\mathbf{t})$'nin etrafında ε yarıçaplı bir tüp veta band tanımlanır. Eğer $\hat{y}(\mathbf{t})$ değeri ε tüpünün içerisinde yer alırsa kayıp yok demektir. Bir diğer deyişle tahmin edilen $\hat{y}(\mathbf{t})$ ile ölçülen değer $y(\mathbf{t})$ arasındaki fark ε'dan az ise kayıp sıfırdır. Tüpün dışında yer alan diğer tüm tahmin noktaları için kayıp, tahmin noktası ile ε yarıçapının farkının mutlak değerine eşittir. ε-SVR algoritması, bağlanım modelini kullanarak bir optimizasyon problemi olarak denklem (4.29)'daki gibi ifade eder.

$$\begin{aligned} &\min_{\mathbf{w},b,\boldsymbol{\xi},\boldsymbol{\xi}^*} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) \\ &Kıs: y_i - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{t}_i) - b \leq \varepsilon + \xi_i, \ \ i = 1, \dots, N \\ &\mathbf{w}^T\boldsymbol{\phi}(\mathbf{t}_i) + b - y_i \leq \varepsilon + \xi_i^*, \ \ i = 1, \dots, N \\ &\xi_i, \xi_i^* \geq 0 \ \ \ i = 1, \dots, N \end{aligned} \tag{4.29}$$

$$\begin{aligned} &\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\beta}, \boldsymbol{\beta}^*, \boldsymbol{\mu}, \boldsymbol{\mu}^*) \\ &= \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N}(\xi_i + \xi_i^*) - \sum_{i=1}^{N}\beta_i[\varepsilon + \xi_i - y_i + \mathbf{w}^T\boldsymbol{\phi}(\mathbf{t}_i) + b] \\ &- \sum_{i=1}^{N}\beta_i^*[\varepsilon + \xi_i^* - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{t}_i) - b + y_i] - \sum_{i=1}^{N}\mu_i\xi_i - \sum_{i=1}^{N}\mu_i^*\xi_i^* \end{aligned} \tag{4.30}$$

$$\begin{aligned} &= \frac{1}{2}\mathbf{w}^T\mathbf{w} - \mathbf{w}^T\sum_{i=1}^{N}(\beta_i - \beta_i^*)\,\boldsymbol{\phi}(\mathbf{t}_i) - \varepsilon\sum_{i=1}^{N}(\beta_i + \beta_i^*) + \sum_{i=1}^{N}y_i(\beta_i - \beta_i^*) \\ &+ b\sum_{i=1}^{N}(\beta_i^* - \beta_i) + \sum_{i=1}^{N}\xi_i(C - \beta_i - \mu_i) + \sum_{i=1}^{N}\xi_i^*(C - \beta_i^* - \mu_i^*) \end{aligned}$$

Denklem (4.30)'da Lagrange çarpanları $\beta_i, \beta_i^*, \mu_i$ ve μ_i^*'lerdir. Bu Lagrange ifadesinin birincil değişkenlere göre türevi alınıp sıfıra eşitlenirse denklem (4.31) elde edilir.

$$\begin{aligned}
&\frac{\partial \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\beta}, \boldsymbol{\beta}^*, \boldsymbol{\mu}, \boldsymbol{\mu}^*)}{\partial \mathbf{w}} = 0 \rightarrow \mathbf{w} = \sum_{i=1}^{N} (\beta_i - \beta_i^*) \boldsymbol{\phi}(\mathbf{t}_i) \\
&\frac{\partial \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\beta}, \boldsymbol{\beta}^*, \boldsymbol{\mu}, \boldsymbol{\mu}^*)}{\partial b} = 0 \rightarrow \sum_{i=1}^{N} (\beta_i - \beta_i^*) = 0 \\
&\frac{\partial \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\beta}, \boldsymbol{\beta}^*, \boldsymbol{\mu}, \boldsymbol{\mu}^*)}{\partial \xi_i} = 0 \rightarrow C - \beta_i - \mu_i = 0, \;\; i = 1, \dots, N \\
&\frac{\partial \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}, \boldsymbol{\xi}^*, \boldsymbol{\beta}, \boldsymbol{\beta}^*, \boldsymbol{\mu}, \boldsymbol{\mu}^*)}{\partial \xi_i^*} = 0 \rightarrow C - \beta_i^* - \mu_i^* = 0, \;\; i = 1, \dots, N
\end{aligned} \tag{4.31}$$

Burada Lagrange çarpanlarının $\boldsymbol{\beta}, \boldsymbol{\beta}^*, \boldsymbol{\mu}, \boldsymbol{\mu}^* \geq \mathbf{0}$ şartlarını sağlamaları gerektiği unutulmamalıdır. Denklem (4.31)'de bulunan eşitlikler Lagrange ifadesinde yerine koyulduğunda,

$$\begin{aligned}
\mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\beta}^*) = &-\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\beta_i - \beta_i^*)(\beta_j - \beta_j^*) \boldsymbol{\phi}^T(\mathbf{t}_i) \boldsymbol{\phi}(\mathbf{t}_j) - \varepsilon \sum_{i=1}^{N} (\beta_i + \beta_i^*) \\
&+ \sum_{i=1}^{N} y_i (\beta_i - \beta_i^*)
\end{aligned} \tag{4.32}$$

elde edilir. Böylece, bağlanım probleminin dual formülasyonu denklem (4.33)'teki gibi ifade edilebilir.

$$\begin{aligned}
\min_{\boldsymbol{\beta}, \boldsymbol{\beta}^*} \mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\beta}^*) = &\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} (\beta_i - \beta_i^*)(\beta_j - \beta_j^*) \boldsymbol{\phi}^T(\mathbf{t}_i) \boldsymbol{\phi}(\mathbf{t}_j) + \varepsilon \sum_{i=1}^{N} (\beta_i + \beta_i^*) \\
&- \sum_{i=1}^{N} y_i (\beta_i - \beta_i^*)
\end{aligned} \tag{4.33}$$

$$Kıs: \sum_{i=1}^{N} (\beta_i - \beta_i^*) = 0$$

$$0 \leq \beta_i, \beta_i^* \leq C, \;\; i = 1, \ldots, N$$

Bu bir karesel programlama (QP) problemidir ve bu problemin çözümü ile tüm β_i, β_i^* elde edilir. $\mathbf{w} = \sum_{i=1}^{N}(\beta_i - \beta_i^*)\boldsymbol{\phi}(\mathbf{t}_i)$ olduğu hatırlanırsa, şeklinde olan bağlanım modeli artık dual formda denklem (4.34) gibi ifade edilir.

$$\hat{y}(\mathbf{t}) = \sum_{i=1}^{N}(\beta_i - \beta_i^*)\boldsymbol{\phi}^T(\mathbf{t}_i)\boldsymbol{\phi}(\mathbf{t}) + b \tag{4.34}$$

İkincil formdaki bağlanım modelinde görülen ve öznitelik uzayında bir iç çarpım olan $\boldsymbol{\phi}^T(\mathbf{t}_i)\boldsymbol{\phi}(\mathbf{t})$ ifadesi dönüşüm fonksiyonuna karşı düşen kernel fonksiyonu ile

$$\boldsymbol{\phi}^T(\mathbf{t}_i)\boldsymbol{\phi}(\mathbf{t}) = Q(\mathbf{t}_i, \mathbf{t}) \tag{4.35}$$

şeklinde ifade edilebileceği için bağlanım modeli denklem (4.36)'daki gibi yazılabilir.

$$\hat{y}(\mathbf{t}) = \sum_{i=1}^{N}(\beta_i - \beta_i^*)Q(\mathbf{t}_i, \mathbf{t}) + b \tag{4.37}$$

QP probleminin çözümünden elde edilen β_i, β_i^* değerleri, $\theta_i = \beta_i - \beta_i^*$ şeklinde tek bir değişken ile ifade edildiğinde model denklem (4.37) gibi yazılır (Vapnik 1995, Vapnik 1998[a]).

$$\hat{y}(\mathbf{t}) = \sum_{i=1}^{N}\theta_i Q(\mathbf{t}_i, \mathbf{t}) + b \tag{4.37}$$

5. ÇEVRİMİÇİ DESTEK VEKTÖR BAĞLANIMI

Destek vektör algoritmaları, eğitim datalarının tamamının bir bütün olarak kullanılmasını zorunlu kılmaktadır. Çevrimiçi zaman serileri benzeri data grupları için bu tür algoritmalar kullanılmamaktadır. Çevrimdışı algoritmalarla gelen her yeni data için algoritmanın baştan çalıştırılıp değerlerin elde edilmesi ise aşırı masraflıdır. Bundan dolayı çevrimiçi destek vektör bağlanımı algoritmasına ihtiyaç duyulmaktadır.

5.1 Karush-Kuhn-Tucker Koşulları

Lagrange çarpanları $\delta_i, \delta_i^*, u_i, u_i^*$ ve ζ olmak üzere, dual optimizasyon problemi denklem (5.1)'deki gibidir (Parrella 2007).

$$\begin{aligned} L_D = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} Q_{ij}(\beta_i - \beta_i^*)(\beta_j - \beta_j^*) - \sum_{i=1}^{N} y_i(\beta_i - \beta_i^*) + \sum_{i=1}^{N} \varepsilon(\beta_i + \beta_i^*) \\ - \sum_{i=1}^{N}(\delta_i\beta_i + \delta_i^*\beta_i^*) + \zeta\sum_{i=1}^{N}(\beta_i - \beta_i^*) \\ - \sum_{i=1}^{N}(u_i(\beta_i - C) + u_i^*(\beta_i^* - C)) \end{aligned} \tag{5.1}$$

$$Kıs: \delta_i, \delta_i^*, u_i, u_i^*, \zeta \geq 0, \;\; i = 1, \dots, N$$

Burada Q_{ij} kernel fonksiyonu olduğunu belirtelim. Bu Lagrange ifadesinin birincil değişkenlere göre türevi alınıp sıfıra eşitlenirse denklem (5.2)'deki eşitlikler elde edilir.

$$\frac{\partial L_D}{\partial \beta_i} = 0 \rightarrow \frac{1}{2}\sum_{i=1}^{N} Q_{ij}(\beta_i - \beta_i^*) + \varepsilon - y_i + \zeta - \delta_i + u_i = 0 \tag{5.2}$$

$$\frac{\partial L_D}{\partial \beta_i^*} = 0 \rightarrow -\frac{1}{2}\sum_{i=0}^{N} Q_{ij}(\beta_i - \beta_i^*) + \varepsilon + y_i - \zeta - \delta_i^* + u_i^* = 0$$

$$\delta_i, \delta_i^* \geq 0 \quad \delta_i \beta_i = 0 \quad \delta_i^* \beta_i^* = 0$$

$$u_i, u_i^* \geq 0 \quad u_i(\beta_i - C) = 0 \quad u_i^*(\beta_i^* - C) = 0$$

KKT koşullarına göre, β_i ve β_i^* değişkenlerinden en az bir tanesi sıfırdan farklı ve pozitif olmalıdır. Bundan dolayı

$$\theta_i = \beta_i - \beta_i^* \tag{5.3}$$

olacak şekilde θ_i gibi bir katsayı tanımlarsak, θ_i katsayısı β_i ve β_i^*'ın her ikisini de belirler. i. data $\mathbf{t}_i$ için,

$$h(\mathbf{t}_i) = f(\mathbf{t}_i) - y_i = \sum_{i=1}^{N} Q_{ij}\theta_j - y_i + b \tag{5.4}$$

şeklinde bir $h(\mathbf{t}_i)$ hata fonksiyonu tanımlarsak

$$\begin{cases} h(\mathbf{t}_i) \geq \varepsilon, & \theta_i = -C \\ h(\mathbf{t}_i) = \varepsilon, & -C < \theta_i < 0 \\ -\varepsilon \leq h(\mathbf{t}_i) \leq \varepsilon, & \theta_i = 0 \\ h(\mathbf{t}_i) = -\varepsilon, & 0 < \theta_i < C \\ h(\mathbf{t}_i) \leq -\varepsilon, & \theta_i = C \end{cases} \tag{5.5}$$

şeklinde bir tablo önümüze çıkar (Parrella 2007).

Eğitim kümesindeki herhangi bir eğitim verisi $\mathbf{t}_i$, sahip olduğu θ_i değerine göre aşağıdaki üç kümeden birine aittir.

$$\begin{gathered} \textit{Hata kümesi: } E = \{i||\theta_i| = C\}\textit{'den} \\ \textit{Destek vektörü kümesi: } S = \{i|0 < |\theta_i| < C\}\textit{'den} \\ \textit{Kalan kümesi: } R = \{i||\theta_i| = 0\} \end{gathered} \tag{5.6}$$

5.2 Artım Algoritması

Artım algoritması, eğitim data setine yeni örnek eklendiğinde eğitilmiş SVR fonksiyonunu güncellemektedir. Yöntem eğitim kümesi data grubunda bulunan örnekler için her adımda KKT koşulları sağlandığından emin olunarak yeni örnek $\mathbf{t}_c$ 'ye uygun θ_c katsayısının KKT koşulları sağlanana kadar değiştirilmesidir (Parrella 2007).

5.2.1 Yeni Örnek Ekleme

Hata fonksiyonu denklem (5.4)'te belirtilmişti. Yeni bir *c* örneği eklendiğinde hata fonksiyonu, denklem (5.7)'deki gibi olacaktır.

$$h(\mathbf{t}_i) = \sum_{j=1}^{N} Q_{ij}\theta_j' + Q_{ic}\theta_c' - b' - y_i \tag{5.7}$$

$$\begin{gathered} \Delta\theta_i = \theta_i' - \theta_i \\ \Delta b = b' - b \\ \Delta h(\mathbf{t}_i) = \sum_{j=1}^{N} Q_{ij}\Delta\theta_j + Q_{ic}\Delta\theta_c + \Delta b \end{gathered} \tag{5.8}$$

$\sum_{i=1}^{N}\theta_i = 0$ özelliği kullanılarak denklem (5.9) elde edilir.

$$\sum_{i=1}^{N}\theta_i = 0 \; ve \; \theta_c + \sum_{i=1}^{N}\theta_i' = 0 \rightarrow \sum_{i=1}^{N}\Delta\theta_i + \Delta\theta_c = 0 \rightarrow \sum_{i=1}^{N}\Delta\theta_i = -\Delta\theta_c \tag{5.9}$$

Bu üç küme içerisinde sadece destek vektörü kümesinde bulunan örnekler, θ_j değerini değiştirebilir. Dolayısıyla sadece destek vektörü kümesi örnekleri için denklem (5.10) yazılabilir.

$$\sum_{j=1}^{N} Q_{ij}\Delta\theta_j + \Delta b = -Q_{ic}\Delta\theta_c, \quad i \in destek\ vektörleri\ kümesi$$
$$\sum_{j\in\mathcal{S}} \Delta\theta_j = -\theta_c \tag{5.10}$$

Eğer destek vektörü kümesi örneklerin indeksleri

$$\mathcal{S} = \{s_1, s_2, s_3, \dots, s_{N_s}\} \tag{5.11}$$

olarak tanımlanırsa denklem (5.10) martis formunda denklem (5.12) gibi yazılabilir.

$$\begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & Q_{s_1s_1} & \cdots & Q_{s_1s_{N_s}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & Q_{s_{N_s}s_1} & \cdots & Q_{s_{N_s}s_{N_s}} \end{bmatrix} \begin{bmatrix} \Delta b \\ \Delta\theta_{s_1} \\ \vdots \\ \Delta\theta_{s_{N_s}} \end{bmatrix} = -\begin{bmatrix} 1 \\ Q_{s_1c} \\ \vdots \\ Q_{s_{N_s}c} \end{bmatrix} \Delta\theta_c$$
$$\begin{bmatrix} \Delta b \\ \Delta\theta_{s_1} \\ \vdots \\ \Delta\theta_{s_{N_s}} \end{bmatrix} = -\begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & Q_{s_1s_1} & \cdots & Q_{s_1s_{N_s}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & Q_{s_{N_s}s_1} & \cdots & Q_{s_{N_s}s_{N_s}} \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ Q_{s_1c} \\ \vdots \\ Q_{s_{N_s}c} \end{bmatrix} \Delta\theta_c \tag{5.12}$$

R matrisi ve $\mathcal{B}$ vektörü denklem (5.13) ve denklem (5.14)'te tanıtılmıştır.

$$\mathbf{R} = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & Q_{s_1s_1} & \cdots & Q_{s_1s_{N_s}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & Q_{s_{N_s}s_1} & \cdots & Q_{s_{N_s}s_{N_s}} \end{bmatrix}^{-1} \tag{5.13}$$

$$\mathcal{B} = \begin{bmatrix} \mathcal{B}_b \\ \mathcal{B}_{s_1} \\ \vdots \\ \mathcal{B}_{s_{N_s}} \end{bmatrix} = -\mathbf{R}\begin{bmatrix} 1 \\ Q_{s_1c} \\ \vdots \\ Q_{s_{N_s}c} \end{bmatrix} \tag{5.14}$$

$$\begin{bmatrix} \Delta b \\ \Delta\theta_{s_1} \\ \vdots \\ \Delta\theta_{s_{N_s}} \end{bmatrix} = \mathcal{B}\Delta\theta_c = \begin{bmatrix} \mathcal{B}_b \\ \mathcal{B}_{s_1} \\ \vdots \\ \mathcal{B}_{s_{N_s}} \end{bmatrix} \Delta\theta_c \tag{5.15}$$

Çözüm, denklem (5.15) gibi yazılabilir (Cauwenberghs ve Poggio 2001). Destek vektörü kümesi örnekleri için $h(\mathbf{t}_i)$ değerinin güncellenmesine gerek yoktur, çünkü her zaman bu değer $|\varepsilon|$'a eşittir.

Hata ve kalan kümesinin örnekleri için bu durum farklıdır. Bu kümedeki örnekler $h(\boldsymbol{t}_i)$ değerini değiştirir. $N = E \cup R = \{n_1, n_2, \dots, n_{N_n}\}$ olacak şekilde bir N vektörü tanımlayalım. (5.8) denklemi, matris formunda yazılırsa, (5.16)'daki eşitlikler elde edilir.

$$\begin{bmatrix} \Delta h(\mathbf{t}_{n_1}) \\ \vdots \\ \vdots \\ \Delta h(\mathbf{t}_{n_{N_n}}) \end{bmatrix} = \begin{bmatrix} \Delta Q_{n_1 c} \\ \vdots \\ \vdots \\ \Delta Q_{n_{N_n} c} \end{bmatrix} \Delta\theta_c + \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & Q_{n_1 s_1} & \cdots & Q_{n_1 s_{N_s}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & Q_{n_{N_n} s_1} & \cdots & Q_{n_{N_n} s_{N_s}} \end{bmatrix} \begin{bmatrix} \Delta b \\ \Delta\theta_{s_1} \\ \vdots \\ \Delta\theta_{s_{N_s}} \end{bmatrix}$$

$$\begin{bmatrix} \Delta h(\mathbf{t}_{n_1}) \\ \vdots \\ \vdots \\ \Delta h(\mathbf{t}_{n_{N_n}}) \end{bmatrix} = \begin{bmatrix} \Delta Q_{n_1 c} \\ \vdots \\ \vdots \\ \Delta Q_{n_{N_n} c} \end{bmatrix} \Delta\theta_c + \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & Q_{n_1 s_1} & \cdots & Q_{n_1 s_{N_s}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & Q_{n_{N_n} s_1} & \cdots & Q_{n_{N_n} s_{N_s}} \end{bmatrix} \mathcal{B}\Delta\theta_c \tag{5.16}$$

$$\gamma = \begin{bmatrix} \Delta Q_{n_1 c} \\ \vdots \\ \vdots \\ \Delta Q_{n_{N_n} c} \end{bmatrix} + \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & Q_{n_1 s_1} & \cdots & Q_{n_1 s_{N_s}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & Q_{n_{N_n} s_1} & \cdots & Q_{n_{N_n} s_{N_s}} \end{bmatrix} \mathcal{B} \tag{5.17}$$

$$\begin{bmatrix} \Delta h(\mathbf{t}_{n_1}) \\ \vdots \\ \vdots \\ \Delta h(\mathbf{t}_{n_{N_n}}) \end{bmatrix} = \gamma\Delta\theta_c \tag{5.18}$$

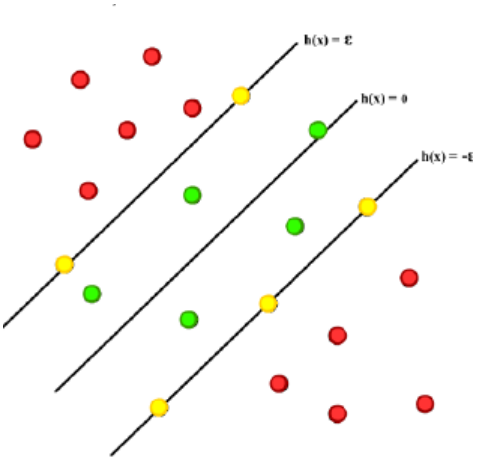
$\mathcal{B}$ 'nın hesaplanmasıyla $\Delta\theta_i$ ve Δb değerleri, γ değerinin hesaplanmasıyla $\Delta h(\mathbf{t}_i)$ değeri güncellenir.

Destek vektörü kümesinin boş olduğu durumda güncellemeler denklem (5.19)'daki gibi olur.

$$\begin{gathered} \Delta b = \Delta\theta_c \\ \Delta h(\mathbf{t}_i) = \Delta b, \;\; i = 1, \dots, N \end{gathered} \tag{5.19}$$

Sonraki adım, eğitim kümesindeki her örnek için $\Delta\theta_c$ sınırı belirlemektir.

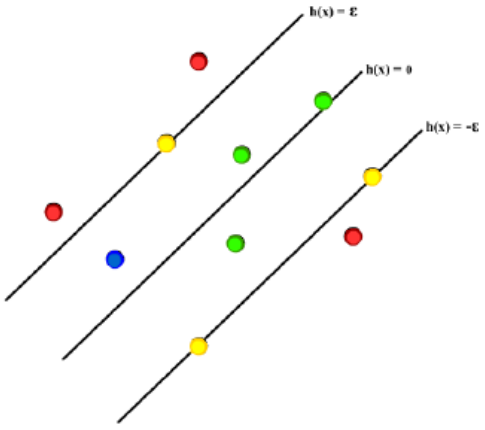
Şekil 5-1 destek vektörü kümesi, kalan kümesi ve hata kümesinin hata fonksiyonu $h(\mathbf{t})$ ile arasındaki ilişkiyi göstermektedir (Parrella 2007).



Şekil 5-1: Destek vektörü kümesi, kalan kümesi ve hata kümesi

Şekil 5-1'de her daire bir örneği temsil etmektedir. Kırmızı daireler hata kümesi örneklerini, sarı daireler destek vektörü kümesi örneklerini ve yeşil daireler kalan kümesi örnekleri göstermektedir.

Yeni bir örnek (mavi daire) eklendiğinde iki farklı durum ortaya çıkar. İlk durum; yeni örneğin hatasının ε değerinden küçük olması durumudur. Bu durumda yeni örnek Şekil 5-2'de görüldüğü gibi kalan kümesine eklenir, diğer örneklerde herhangi bir değişiklik yapılmaz (Parrella 2007).



Şekil 5-2: Yeni örneğin kalan kümesine eklendiği durum

İkinci durum ise, yeni örneğin hatasının ε değerinden büyük olduğu durumdur. Bu durumda yeni örneğin θ_c veya $h(\mathbf{t}_c)$ değeri, destek vektörü kümesi ya

da hata kümesine ulaşıncaya kadar değiştirilir. Yeni örneği destek vektörü kümesi ya da hata kümesine taşımadan önce tüm örnekler en az farkla kümelerini değiştirene kadar hareket ettirilir. Bir diğer önemli nokta da tüm örnekler yeni örnekle aynı yönde hareket etmeyebilir. Destek vektörü kümesindeki örnekler için yönü $\mathcal{B}_i$ değeri, hata kümesi ya da kalan kümesindeki örnekler içinse γ_i değeri belirler.

5.2.1.1 Destek Vektörü Kümesinden Hata Kümesine Olan Hareket

$\mathbf{t}_i$ örneği destek vektörü kümesinden hata kümesine hareket ettirilirken; θ_i değeri $0 < |\theta_i| < C$ iken $|\theta_i| = C$; $|h(\mathbf{t}_i)| = \varepsilon$ iken $|h(\mathbf{t}_i)| > \varepsilon$ olur. Destek vektörü kümesi örnekleri için güncelleme formülleri eşitlik (5.20)'de verilmiştir.

$$\begin{aligned} \Delta\theta_i &= \mathcal{B}_i \Delta\theta_c \\ \Delta\theta_c &= \Delta\theta_i / \mathcal{B}_i \\ \Delta\theta_c &= (C - \theta_i)/\mathcal{B}_i \ \ veya \ \ (-C - \theta_i)/\mathcal{B}_i \end{aligned} \tag{5.20}$$

5.2.1.2 Destek Vektörü Kümesinden Kalan Kümesine Olan Hareket

$\mathbf{t}_i$ örneği destek vektörü kümesinden kalan kümesine hareket ettirilirken; θ_i değeri $0 < |\theta_i| < C$ iken $|\theta_i| = 0$; $|h(\mathbf{t}_i)| = \varepsilon$ iken $|h(\mathbf{t}_i)| < \varepsilon$ olur. Destek vektörü kümesi örnekleri için güncelleme formülleri eşitlik (5.21)'de verilmiştir.

$$\begin{aligned} \Delta\theta_i &= \mathcal{B}_i \Delta\theta_c \\ \Delta\theta_c &= \Delta\theta_i / \mathcal{B}_i \\ \Delta\theta_c &= -\theta_i / \mathcal{B}_i \end{aligned} \tag{5.21}$$

5.2.1.3 Hata Kümesinden Destek Vektörü Kümesine Olan Hareket

$\mathbf{t}_i$ örneği hata kümesinden destek vektörü kümesine hareket ettirilirken; $|h(\mathbf{t}_i)| > \varepsilon$ iken $|h(\mathbf{t}_i)| = \varepsilon$ olur. Hata kümesi örnekleri için güncelleme formülleri eşitlik (5.22)'de verilmiştir.

$$\begin{aligned} \Delta h(\mathbf{t}_i) &= \gamma_i \Delta\theta_c \\ \Delta\theta_c &= \Delta h(\mathbf{t}_i)/\gamma_i \\ \Delta\theta_c &= (-\varepsilon - h(\mathbf{t}_i))/\gamma_i \ \ veya \ \ (\varepsilon - h(\mathbf{t}_i))/\gamma_i \end{aligned} \tag{5.22}$$

5.2.1.4 Kalan Kümesinden Destek Vektörü Kümesine Olan Hareket

$\mathbf{t}_i$ örneği kalan kümesinden destek vektörü kümesine hareket ettirilirken; $|h(\mathbf{t}_i)| < \varepsilon$ iken $|h(\mathbf{t}_i)| = \varepsilon$ olur. Hata kümesi örnekleri için güncelleme formülleri eşitlik (5.23)'te verilmiştir.

$$\begin{aligned} \Delta h(\mathbf{t}_i) &= \gamma_i \Delta\theta_c \\ \Delta\theta_c &= \Delta h(\mathbf{t}_i)/\gamma_i \\ \Delta\theta_c &= (-\varepsilon - h(\mathbf{t}_i))/\gamma_i \ \ veya \ \ (\varepsilon - h(\mathbf{t}_i))/\gamma_i \end{aligned} \tag{5.23}$$

5.2.2 Algortima

5.2.2.1 Girişler ve Çıkışlar

Girişler;

- Eğitim seti $\{\mathbf{t}_i, y_i, i = 1, \dots, N\}$
- Ağırlıklar $\theta_i, i = 1, \dots, N$
- Eşik değeri b
- Destek vektörü kümesi, kalan kümesi, hata kümesi

- Parametreler ε, C ve Kernel parametreleri
- **R** matrisi
- Yeni örnek $c = (\mathbf{t}_c, y_c)$

Eğitim işleminin başlangıcında, Eğitim kümesi, destek vektörü kümesi, kalan kümesi, hata kümesi, R matrisi boş küme; eşik değeri ise sıfır olarak belirlenir.

Çıkışlar;

- Yeni eğitim kümesi $\{\mathbf{t}_i, y_i, i = 1, \dots, N + 1\}$
- Yeni ağırlıklar $\theta_i, i = 1, \dots, N + 1$
- Yeni eşik değeri b
- Yeni destek vektörü kümesi, kalan kümesi, hata kümesi
- Yeni **R** matrisi

Çıkışlar, giriş değerlerinin güncellenmiş halini içerir.

5.2.2.2 Eğitim Algoritması

i) Yeni örneği $(\mathbf{t}_c, y_c)$ eğitim setine ekle

ii) $\theta_c = 0$

iii) $f(\mathbf{t}_c)$ ve $h(\mathbf{t}_c)$ değerlerini hesapla.

iv) $h(\mathbf{t}_c) < \varepsilon$ ise

- Yeni örneği kalan kümesine ekle ve çık.

v) $h(\mathbf{t}_i)$ değerini hesapla.

vi) Yeni örnek eklenmediği sürece devam et.

- $\mathcal{B}$ ve γ değerlerini güncelle.
- En küçük farkları hesapla $(L_{c1}, L_{c2}, \mathbf{L}_S, \mathbf{L}_E, \mathbf{L}_R)$
- $\Delta\theta_c = \min(L_{c1}, L_{c2}, \mathbf{L}_S, \mathbf{L}_E, \mathbf{L}_R)$
- $L_{c1} = 1, L_{c2} = 2, \mathbf{L}_S = 3, \mathbf{L}_E = 4, \mathbf{L}_R = 5$
- θ_c, θ_i, b değerlerini güncelle.
- $h(\mathbf{t}_i), i \in E \cup R$ değerini güncelle.
  - Durum 1
    - Yeni örneği destek vektörü kümesine ekle.

- Yeni örneği **R** matrisine ekle.
- Çık.

➢ Durum 2

- Yeni örneği hata kümesine ekle.
- Çık

➢ Durum 3

- $\theta_l = 0$ ise
  - l örneğini, destek vektörü kümesinden kalan kümesine taşı.
  - l örneğini, **R** matrisinden sil.
- $\theta_l = |C|$ ise
  - l örneğini destek vektörü kümesinden hata kümesine taşı.
  - l örneğini, **R** matrisinden sil.

➢ Durum 4

- l örneğini hata kümesinden destek vektörü kümesine taşı.
- l örneğini **R** matrisine ekle.

➢ Durum 5

- l örneğini kalan kümesinden destek vektörü kümesine taşı.
- l örneğini **R** matrisine ekle.

İlk olarak, Çevrimiçi Destek Vektör Bağlanımı yeni örneğin kalan kümesine yerleşip yerleşmeyeceğini kontrol eder. Yeni örnek kalan kümesine yerleşmezse, bir döngü başlar ve bu döngü yeni örnek hata kümesi ya da destek vektörü kümesine yerleşinceye kadar devam eder (Parrella 2007).

5.2.2.3 En Az Farkın Bulunması

Mümkün olan tüm hareketleri analiz etmeden önce, hareket yönünü tanımlamak gerekir. Artım algoritması için yön, $h(\mathbf{t}_c)$ değerinin ters işaretlisi alınır. Çünkü amaç $h(\mathbf{t}_c)$ değerini $|\varepsilon|$'a ulaşıncaya kadar azaltmaktır.

$$Yön = sign(-h(\mathbf{t}_c)) \tag{5.24}$$

5.2.2.3.1 L_{c1} Farkı

Yeni örneğin $h(\mathbf{t}_c)$ değerinin $|\varepsilon|$' ulaştığı ve dolayısıyla destek vektörü kümesine eklendiği durumda hesaplanır.

i. $((\theta_c > 0) \cup (\theta_c = 0 \cap h(\mathbf{t}_c) < 0)$ ise
   - $L_{c1} = (-h(\mathbf{t}_c) - \varepsilon)/\gamma_c$

ii. Değilse
   - $L_{c1} = (-h(\mathbf{t}_c) + \varepsilon)/\gamma_c$

5.2.2.3.2 L_{c2} Farkı

Yeni örneğin hata kümesine olan uzaklığını ölçer.

i. $Yön > 0$ ise
   - $L_{c2} = -\theta_c + C$

ii. Değilse
   - $L_{c2} = -\theta_c - C$

5.2.2.3.3 $\mathbf{L}_S$ Farkı

Her bir destek vektörü kümesi örneğinin kalan kümesine ya da hata kümesine olan uzaklığını hesaplar. Örneklerin yönü, yeni örneğin yönü ve $\mathcal{B}_i$ değerine göre değişir.

i. Eğer $Yön \times \mathcal{B}_i > 0$ ise
   - Eğer $h(\mathbf{t}_i) > 0$
     - Eğer $\theta_i < -C$
       - $\mathbf{L}_{S_i} = (-\theta_i - C)/\mathcal{B}_i$
     - Eğer $\theta_i \leq 0$
       - $\mathbf{L}_{S_i} = -\theta_i/\mathcal{B}_i$
     - Eğer $\theta_i > 0$
       - $\mathbf{L}_{S_i} = Yön \times inf$

- Eğer $h(\mathbf{t}_i) = -\varepsilon$
  - Eğer $\theta_i < 0$
    - $\mathbf{L}_{S_i} = (-\theta_i + C)/\mathcal{B}_i$
  - Eğer $\theta_i > C$
    - $\mathbf{L}_{S_i} = Yön \times inf$

ii. Eğer $Yön \times \mathcal{B}_i < 0$ ise

- Eğer $h(\mathbf{t}_i) > 0$ ise
  - Eğer $\theta_i < -C$
    - $\mathbf{L}_{S_i} = Yön \times inf$
  - Eğer $\theta_i \leq 0$
    - $\mathbf{L}_{S_i} = (-\theta_i - C)/\mathcal{B}_i$
  - Eğer $\theta_i > 0$ ise
    - $\mathbf{L}_{S_i} = -\theta_i/\mathcal{B}_i$
- Eğer $h(\mathbf{t}_i) = \varepsilon$
  - Eğer $\theta_i < 0$ ise
    - $\mathbf{L}_{S_i} = Yön \times inf$
  - Eğer $\theta_i \leq C$
    - $\mathbf{L}_{S_i} = -\theta_i/\mathcal{B}_i$
  - Eğer $\theta_i > C$
    - $\mathbf{L}_{S_i} = (-\theta_i + C)/\mathcal{B}_i$

5.2.2.3.4 $\mathbf{L}_E$ Farkı

Hata kümesindeki her bir örneğin destek vektörü kümesine olan uzaklığını hesaplar. Örneklerin yönü, yeni örneğin yönü ve γ_i değerine göre değişir.

i. Eğer $Yön \times \gamma_i > 0$ ise

- Eğer $\theta_i > 0$
  - Eğer $h(\mathbf{t}_i) < -\varepsilon$
    - $\mathbf{L}_{E_i} = (-h(\mathbf{t}_i) - \varepsilon)/\gamma_i$
  - Eğer $h(\mathbf{t}_i) \geq -\varepsilon$
    - $\mathbf{L}_{E_i} = Yön \times inf$

- Eğer $\theta_i = -C$
  - Eğer $h(\mathbf{t}_i) < \varepsilon$
    - $\mathbf{L}_{E_i} = (-h(\mathbf{t}_i) + \varepsilon)/\gamma_i$
  - Eğer $h(\mathbf{t}_i) \geq \varepsilon$
    - $\mathbf{L}_{E_i} = Yön \times inf$

ii. Eğer $Yön \times \gamma_i < 0$ ise

- Eğer $\theta_i > 0$ ise
  - Eğer $h(\mathbf{t}_i) > -\varepsilon$
    - $\mathbf{L}_{E_i} = (-h(\mathbf{t}_i) - \varepsilon)/\gamma_i$
  - Eğer $h(\mathbf{t}_i) \leq -\varepsilon$
    - $\mathbf{L}_{E_i} = Yön \times inf$
- Eğer $\theta_i = -C$
  - Eğer $h(\mathbf{t}_i) > \varepsilon$ ise
    - $\mathbf{L}_{E_i} = (-h(\mathbf{t}_i) + \varepsilon)/\gamma_i$
  - Eğer $h(\mathbf{t}_i) \leq \varepsilon$
    - $\mathbf{L}_{E_i} = Yön \times inf$

5.2.2.3.5 $\mathbf{L}_R$ Farkı

Kalan kümesindeki her bir örneğin destek vektörü kümesine olan uzaklığını hesaplar. Örneklerin yönü, yeni örneğin yönü ve γ_i değerine göre değişir.

i. Eğer $Yön \times \gamma_i > 0$

- Eğer $h(\mathbf{t}_i) < -\varepsilon$
  - $\mathbf{L}_{R_i} = (-h(\mathbf{t}_i) - \varepsilon)/\gamma_i$
- Eğer $h(\mathbf{t}_i) < +\varepsilon$
  - $\mathbf{L}_{R_i} = (-h(\mathbf{t}_i) + \varepsilon)/\gamma_i$
- Diğer durumlar
  - $\mathbf{L}_{R_i} = Yön \times inf$

ii. Eğer $Yön \times \gamma_i < 0$

- Eğer $h(\mathbf{t}_i) > \varepsilon$
  - $\mathbf{L}_{R_i} = (-h(\mathbf{t}_i) + \varepsilon)/\gamma_i$

- Eğer $h(\mathbf{t}_i) > -\varepsilon$
  - $\mathbf{L}_{R_i} = (-h(\mathbf{t}_i) - \varepsilon)/\gamma_i$

5.2.2.4 R Matrisinin Etkin Olarak Hesaplanması

(5.14) denkleminde kullanılan **R** matrisi destek vektörü kümesi değiştiğinde güncellenmelidir (Ma ve diğ. 2003).

$$\mathbf{R} = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & Q_{s_1 s_1} & \cdots & Q_{s_1 s_{N_s}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & Q_{s_{N_s} s_1} & \cdots & Q_{s_{N_s} s_{N_s}} \end{bmatrix}^{-1} \tag{5.25}$$

R matrisini, ters matris hesaplamadan etkin bir şekilde güncelleyebiliriz. destek vektörü kümesindeki k. örnek $\mathbf{t}_{s_k}$, destek vektörü kümesinden çıkarıldığında, yeni **R** matrisi denklem (5.26)'daki gibi elde edilir.

$$\begin{aligned} I &= [1 \cdots (k-1)(k+1) \cdots (N_s+1)] \\ \mathbf{R}_{yeni} &= \mathbf{R}_{I,I} - \frac{\mathbf{R}_{I,k}\mathbf{R}_{k,I}}{\mathbf{R}_{k,k}} \end{aligned} \tag{5.26}$$

Destek vektörü kümesine yeni örnek eklendiğinde, yeni **R** matrisi

$$\begin{aligned} \mathbf{R}_{yeni} &= \begin{bmatrix} & \mathbf{R} & & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} \\ 0 & \cdots & 0 & 0 \end{bmatrix} + \frac{1}{\gamma_i}\begin{bmatrix} \mathcal{B} \\ 1 \end{bmatrix}\begin{bmatrix} \mathcal{B}^T & 1 \end{bmatrix} \\ \mathcal{B} &= -\mathbf{R}\begin{bmatrix} 1 \\ Q_{is_1} \\ \vdots \\ Q_{is_{N_s}} \end{bmatrix} \\ \gamma_i &= Q_{ii} + \begin{bmatrix} 1 & Q_{is_1} & \dots & Q_{is_{N_s}} \end{bmatrix}\mathcal{B} \end{aligned} \tag{5.27}$$

şeklinde elde edilebilir. $\mathcal{B}$ ve γ_i değerlerinin hesaplanması $\mathbf{t}_i$ örneğinin hata kümesinden kalan kümesine götürüldüğünde geçerlidir. $\mathbf{t}_i$ örneği destek vektörü

kümesine eklendiğinde; $\mathcal{B}$ denklem (5.14)'e göre, γ_i ise denklem (5.17)'de tanımlanan γ'nın son elemanı olarak elde edilebilir.

5.3 Azaltım Algoritması

Azaltım algoritması, eğitim kümesindeki herhangi bir örnek silindiğinde kullanılmaktadır.

5.3.1 Giriş ve Çıkışlar

Girişler;

- Eğitim kümesi $\{\mathbf{t}_i, y_i, i = 1, \dots, N\}$
- Ağırlıklar $\theta_i, i = 1, \dots, N$
- Eşik değeri b
- Destek vektörü kümesi, kalan kümesi, hata kümesi
- Parametreler ε, C ve Kernel parametreleri
- **R** matrisi
- Silinecek örneğin indeksi (c)

Silinecek örneğin indeksi dışındaki tüm girişler, artım algoritmasının girişleri ile aynıdır.

Çıkışlar;

- Yeni eğitim kümesi $\{\mathbf{t}_i, y_i, i = 1, \dots, N + 1\}$
- Yeni ağırlıklar $\theta_i, i = 1, \dots, N + 1$
- Yeni eşik değeri b
- Yeni destek vektörü kümesi, hata kümesi ve kalan kümesi
- Yeni **R** matrisi

Çıkışlar, giriş değerlerinin güncellenmiş halini içerir.

5.3.2 Algoritma

i. Eğer $c \in kalan\ kümesi$
   - Örneği kalan kümesinden sil.
   - Örneği eğitim setinden sil
   - Çık

ii. Eğer $c \in destek\ vektörü\ kümesi$
   - Örneği destek vektörü kümesinden sil.

iii. Eğer $c \in hata\ kümesi$
   - Örneği hata kümesinden sil.

iv. $h(\mathbf{t}_i)$ değerini hesapla

v. c örneği silinmediği sürece devam et.
   - $\mathcal{B}$ ve γ değerlerini güncelle.
   - En az farkları hesapla. $(L_C, \mathbf{L}_S, \mathbf{L}_E, \mathbf{L}_R)$
   - $\Delta\theta_c = \min(L_C, \mathbf{L}_S, \mathbf{L}_E, \mathbf{L}_R)$
   - $L_C = 1, \mathbf{L}_S = 2, \mathbf{L}_E = 3, \mathbf{L}_R = 4$
   - θ_c, θ_i, b değerlerini güncelle.
   - $h(\mathbf{t}_i), i \in E \cup R$ değerini güncelle.
   - Durum 1
     - Örneği eğitim setinden sil ve çık.
   - Durum 2
     - $\theta_l = 0$ ise
       - l örneğini destek vektörü kümesinden kalan kümesine taşı
       - l örneğini **R** matrisinden sil.
     - $\theta_l = |C|$ ise
       - l örneğini destek vektörü kümesinden hata kümesine taşı
       - l örneğini **R** matrisinden sil.
   - Durum 3
     - l örneğini hata kümesinden destek vektörü kümesine taşı.
     - l örneğini **R** matrisine ekle.

- Durum 4
  - l örneğini kalan kümesinden destek vektörü kümesine taşı.
  - l örneğini **R** matrisine ekle.

İlk olarak, Çevrimiçi destek vektör bağlanımı silinecek örneğin setini kontrol eder. Eğer örnek kalan kümesinde ise silmek kolaydır. Örnek kalan kümesinde değilse bir döngü başlar ve bu döngü silinecek örnek kalan kümesine yerleşinceye kadar yani $\theta_c = 0$ olana kadar devam eder (Parrella 2007).

5.3.3 En Az Farkı Bulma

Örnek silme işlemi boyunca, hareket yönü θ_c değerine bağlıdır.

$$Yön = sign(-\theta_c) \tag{5.28}$$

Azaltım algoritmasında L_{c_1} ve L_{c_2} farkları yerini L_c farkına bırakır.

5.3.3.1 L_c Farkı

L_c farkı θ_c değerinin 0'a olan uzaklığını hesaplar.

$$L_c = -\theta_c \tag{5.29}$$

$\mathbf{L}_S$,$\mathbf{L}_E$ ve $\mathbf{L}_R$ farkları artım algortimasıyla aynıdır. Tek fark hareket yönünün θ_c'ye bağlı olmasıdır.

6. ÇEVRİMİÇİ DESTEK VEKTÖR MAKİNELERİYLE GENELLEŞTİRİLMİŞ ÖNGÖRÜLÜ DENETİM

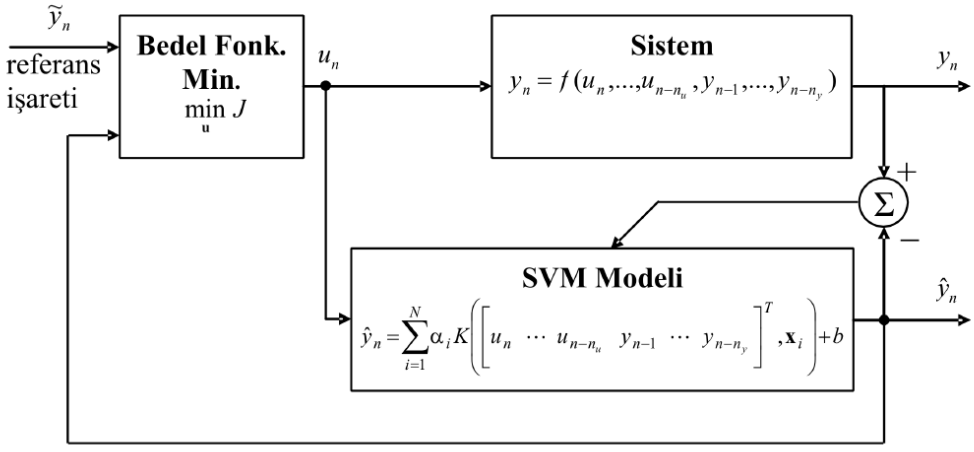
6.1 Model Öngörülü Denetim

Dinamiği denklem (6.1)'deki gibi olan bir NARX (Nonlinear Auto-Regressive eXogenous) modeliyle temsil edilen doğrusal-olmayan sistemi ele alalım.

$$y_n = f(u_n, \dots, u_{n-n_u}, y_{n-1}, \dots, y_{n-n_y}) \tag{6.1}$$

Burada u_n n zaman indeksi anında sisteme uygulanan denetim işareti, y_n ise sistemin buna karşı düşen çıkışıdır, n_u ve n_y ise sırasıyla modelde yer alan geçmiş denetim işareti ve çıkış işareti sayılarıdır. Bu arada doğrusal olmayan f'in bilinmediği varsayılmaktadır.

Şekil 6-1 MPC yapısının denetim döngüsünü göstermektedir, burada $\hat{y}_n$ model çıkışının zaman indeksi anındaki değeri ve $\tilde{y}$ ise sistem tarafından takip edilmesi istenen referans işaretidir (İplikçi 2006[a,b]).



Şekil 6-1: MPC döngüsü

MPC yapısı iki temel elemandan oluşmaktadır. İlki daha sonradan tanımlanacak olan **u** aday denetim vektörüne karşı sistemin üreteceği cevabı tahmin etmede kullanılan sistem modelidir. Bunun yanısıra, sistem modeli, MPC'nin Bedel Fonksiyonu Minimizasyon (*Cost Function Minimization* - *CFM*) bloğu için gerekli eğitim bilgisinin elde edilmesinde kullanılır. CFM bloğunun amacı denklem (6.2)'de verilen başarım göstergesi *J*'nin aday denetim vektörü **u**'ya göre en aza indirilmesidir.

$$J = \sum_{j=N_1}^{N_2} \left(\tilde{y}_{n+j} - \hat{y}_{n+j}\right)^2 + \sum_{j=1}^{N_u} \lambda_j \left(\Delta u_{n+j}\right)^2 + \sum_{j=1}^{N_u} \left(\frac{\mu}{u_{n+j} + \frac{\rho}{2} - \vartheta} + \frac{\mu}{\frac{\rho}{2} - \vartheta - u_{n+j}} - \frac{4}{\rho} \right) \tag{6.2}$$

Burada N_1 en kısa bedel ufku, N_2 en uzun bedel ufku, N_u denetim ufku, λ ağırlık faktörüdür ve Δu_{n+j} ise $\Delta u_{n+j} = u_{n+j} - u_{n+j-1}$ şeklinde verilmiştir.

CFM algoritmasında, denklem (6.3)'te verilen aday denetim vektörü **u**'nun elemanları, denklem (6.4)'te verilen kurala göre güncellenir.

$$\mathbf{u} = \left[u_{n+1}\ u_{n+2} \ldots\ u_{n+N_u}\right]^T \tag{6.3}$$

$$\mathbf{u} \leftarrow \mathbf{u} + s\mathbf{p} \tag{6.4}$$

Burada **p** arama yönü ve *s* ise adım aralığıdır. Her bir örnekleme anında, denetim işareti ve sistem çıkışları üzerindeki kısıtlamalar gözönüne alınarak en uygun arama yönü belirlendikten sonra, bu arama yönüne dayanarak en uygun adım aralığı belirlenir. Ardından, denetim vektörü **u** güncellenerek vektörün ilk elemanı sisteme uygulanır.

En uygun arama yönünü bulmak için, optimizasyon kuramı literatüründe mevcut yöntemler (Yang ve diğ. 2005, Nocedal ve Wright 1999) kullanılır. CFM algoritmasında kullanılan optimizasyon tekniğine bağlı olarak, Taylor açılımında ikinci dereceye kadar olan eğim bilgisine ihtiyaç duyulabilir. Dik İniş (Gradient

Descent ($\mathbf{p} = -\mathbf{g}$)) gibi birinci dereceden arama algoritmaları denklem (6.5)'te verilen eğitim vektörünün hesabını gerektirirken, Değiştirilmiş Newton Yöntemi (Modified Newton's Method ($\mathbf{p} = -\mathbf{H}^{-1}\mathbf{g}$)) gibi ikinci dereceden arama algoritmaları ilave olarak denklem (6.6)'da verilen Hessian matrisinin hesabını gerektirir.

$$\mathbf{g} = \frac{\partial \mathrm{J}}{\partial \mathbf{u}} = \left[\frac{\partial J}{\partial u_{n+1}} \; \frac{\partial J}{\partial u_{n+2}} \; \cdots \frac{\partial J}{\partial u_{n+N_u}}\right]^T \tag{6.5}$$

$$\boldsymbol{H} = \frac{\partial^2 J}{\partial \mathbf{u}^2} = \begin{bmatrix} \frac{\partial^2 J}{\partial u_{n+1} u_{n+1}} & \frac{\partial^2 J}{\partial u_{n+1} u_{n+2}} & \cdots & \frac{\partial^2 J}{\partial u_{n+1} u_{n+N_u}} \\ \frac{\partial^2 J}{\partial u_{n+2} u_{n+1}} & \frac{\partial^2 J}{\partial u_{n+2} u_{n+2}} & \cdots & \frac{\partial^2 J}{\partial u_{n+2} u_{n+N_u}} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 J}{\partial u_{n+N_u} u_{n+1}} & \frac{\partial^2 J}{\partial u_{n+N_u} u_{n+2}} & \cdots & \frac{\partial^2 J}{\partial u_{n+N_u} u_{n+N_u}} \end{bmatrix} \tag{6.6}$$

Eğim g'de, h^{inci} eleman denklem (6.7) ile verilmektedir.

$$\begin{aligned} \frac{\partial J}{\partial u_{n+h}} = -2 \sum_{j=N_1}^{N_2} (\tilde{y}_{n+j} - \hat{y}_{n+j}) \frac{\partial \hat{y}_{n+j}}{\partial u_{n+h}} + 2 \sum_{j=1}^{N_u} \lambda_j (\Delta u_{n+j}) (\delta_{h,j} - \delta_{h,j-1}) \\ + \sum_{j=1}^{N_u} \delta_{h,j} \left(- \frac{\mu}{\left(u_{n+j} + \frac{\rho}{2} - \vartheta\right)^2} + \frac{\mu}{\left(\frac{\rho}{2} + \vartheta - u_{n+j}\right)^2} \right) \end{aligned} \tag{6.7}$$

$$h = 1, \ldots, N$$

Burada $\delta_{i,j}$ Kronecker Delta fonksiyonudur. Benzer şekilde, Hessian matrisi H'nin m^{inci}, h^{inci} elemanı da denklem (6.8)'deki gibidir.

$$\frac{\partial^2 J}{\partial u_{n+m}u_{n+h}} = 2\sum_{j=N_1}^{N_2}\frac{\partial \hat{y}_{n+j}}{\partial u_{n+m}}\frac{\partial \hat{y}_{n+j}}{\partial u_{n+h}} - 2\sum_{j=N_1}^{N_2}\frac{\partial^2 \hat{y}_{n+j}}{\partial u_{n+m}u_{n+h}}(\tilde{y}_{n+j} - \hat{y}_{n+j}) + \sum_{j=1}^{N_u}\lambda_j\left(\delta_{m,j} - \delta_{m,j-1}\right)\left(\delta_{h,j} - \delta_{h,j-1}\right) + \sum_{j=1}^{N_u}\delta_{m,j}\delta_{h,j}\left(-\frac{2\mu}{\left(u_{n+j} + \frac{\rho}{2} - \vartheta\right)^3} + \frac{2\mu}{\left(\frac{\rho}{2} + \vartheta - u_{n+j}\right)^3}\right) \tag{6.8}$$

Denklem (6.7) ve (6.8)'den görüleceği gibi, birinci dereceden terimler $\left(\frac{\partial \hat{y}_{n+j}}{\partial u_{n+h}}\right)$ ve ikinci dereceden terimler $\left(\frac{\partial^2 \hat{y}_{n+j}}{\partial u_{n+m}\partial u_{n+h}}\right)$ her bir adımda pek çok kez hesaplanmaktadır. Elde edilen modele dayalı olarak bu terimlerin hesabının işlemsel yoğunluk anlamında MPC'nin uygulanabilirliği üzerinde doğrudan etkisi vardır.

6.2 Çevrimiçi SVM-Tabanlı MPC

Kontrol edilecek sistemin SVM modeli elde edildikten sonra, bu model, MPC yaklaşımı izlenerek alt bölümlerde verilen formüller ve algoritma yoluyla Çevrimiçi SVM-Tabanlı MPC yapısında kullanılabilir.

6.2.1 SVM Modelinden Eğim Bilgisinin Elde Edilmesi

Şu anki giriş vektörü aşağıdaki biçimde yazılırsa,

$$\mathbf{c}_n = \left[u_n \cdots u_{n-n_u}\, y_n \cdots y_{n-n_y}\right]^T \tag{6.9}$$

SVM modelinin buna karşı düşen çıkışı da denklem (6.10)'daki gibi ifade edilebilir.

$$\hat{y}_n = \sum_{j=1}^{\#SV}\alpha_j K\left(\mathbf{c}_n, \mathbf{t}_j\right) + b \tag{6.10}$$

SVM-Tabanlı MPC'de çekirdek fonksiyonu olarak Radyal Tabanlı Fonksiyon (Radial Basis Function-RBF) kullanılmıştır.

$$K_{ij} = K(\mathbf{t}_i, \mathbf{t}_j) = \exp(-\frac{(\mathbf{t}_i - \mathbf{t}_j)^T(\mathbf{t}_i - \mathbf{t}_j)}{2\sigma^2}) \tag{6.11}$$

Burada σ genişlik parametresidir. Eğer d_{jn}, denklem (6.12)'de verildiği gibi j^{inci} destek vektörü $\mathbf{t}_j$ ile şu anki durum $\mathbf{c}_n$ arasındaki Öklit uzaklığı olarak tanımlanırsa,

$$\begin{aligned} d_{jn} &= (\mathbf{c}_n - \mathbf{t}_j)^T(\mathbf{c}_n - \mathbf{t}_j) \\ &= \sum_{i=0}^{n_u}(t_{j,i+1} - u_{n-i})^2 + \sum_{i=1}^{n_y}(t_{j,n_u+i+1} - y_{n-i})^2 \end{aligned} \tag{6.12}$$

Çekirdek fonksiyonu,

$$K(\mathbf{c}_n, \mathbf{t}_j) = \exp\left(-\frac{d_{jn}}{2\sigma^2}\right) \tag{6.13}$$

şeklinde yeniden yazılabilir ve bağlanım modeli denklem (6.14)'teki gibi ifade edilir.

$$\hat{y}_n = \sum_{j=1}^{\#SV} \alpha_j \exp\left(-\frac{d_{jn}}{2\sigma^2}\right) + b \tag{6.14}$$

Artık, SVM bağlanım modelini kullanarak denklem (6.15)'de verilen formüllerle sistemin gelecekteki davranışını tahmin edebiliriz.

$$\hat{y}_{n+k} = \sum_{j=1}^{\#SV} \alpha_j \exp\left(-\frac{d_{j,n+k}}{2\sigma^2}\right) + b\,, \quad k = N_1, \dots, N_2 \tag{6.15}$$

Burada

$$d_{j,n+k} = \sum_{i=1}^{\min(k,n_y)} \left(t_{n_u+i+1} - \hat{y}_{n+k-i}\right)^2 + \sum_{i=k+1}^{n_y} \left(t_{n_u+i+1} - \hat{y}_{n+k-i}\right)^2 + \sum_{i=0}^{n_u} \begin{cases} \left(t_{j,i+1} - u_{n+k-i}\right)^2, & k - N_u < i \\ \left(t_{j,i+1} - u_{n+N_u}\right)^2, & k - N_u \geq i \end{cases} \tag{6.16}$$

Bu tanımlar kullanılarak, birinci-dereceden kısmi türevler aşağıdaki gibi bulunur:

$$\frac{\partial \hat{y}_{n+k}}{\partial u_{n+h}} = \sum_{j=1}^{\#SV} \alpha_j \frac{\partial \exp\left(-\frac{d_{j,n+k}}{2\sigma^2}\right)}{\partial u_{n+h}} \tag{6.17}$$

Burada

$$\begin{aligned} \frac{\partial \exp\left(-\frac{d_{j,n+k}}{2\sigma^2}\right)}{\partial u_{n+h}} &= \frac{\partial \exp\left(-\frac{d_{j,n+k}}{2\sigma^2}\right)}{\partial d_{j,n+k}} \frac{\partial d_{j,n+k}}{\partial u_{n+h}} \\ &= -\frac{1}{2\sigma^2} \exp\left(-\frac{d_{j,n+k}}{2\sigma^2}\right) \frac{\partial d_{j,n+k}}{\partial u_{n+h}} \end{aligned} \tag{6.18}$$

şeklindedir ve

$$\begin{aligned} \frac{\partial d_{j,n+k}}{\partial u_{n+h}} = -2 & \sum_{i=1}^{\min(k,n_y)} t_{j,n_u+i+1} \frac{\partial \hat{y}_{n+k-i}}{\partial u_{n+h}} \delta_1(k-i-1) \\ & + 2 \sum_{i=1}^{\min(k,n_y)} \hat{y}_{n+k-i} \frac{\partial \hat{y}_{n+k-i}}{\partial u_{n+h}} \delta_1(k-i-1) \\ & - 2 \sum_{i=0}^{n_u} \begin{cases} \left(t_{j,i+1} - u_{n+k-i}\right), & k - N_u < i \\ \left(t_{j,i+1} - u_{n+N_u}\right), & k - N_u \geq i \end{cases} \end{aligned} \tag{6.19}$$

ile verilir ki burada $\delta_1(.)$ Birim basamak fonksiyonudur. İkinci-dereceden kısmi türevler de şu şekilde elde edilir:

$$\frac{\partial^2 \hat{y}_{n+k}}{\partial u_{n+h}\partial u_{n+m}} = \sum_{j=1}^{\#SV} \alpha_j \frac{\partial^2 \exp\left(-\frac{d_{j,n+k}}{2\sigma^2}\right)}{\partial u_{n+h}\partial u_{n+m}} \tag{6.20}$$

burada

$$\begin{aligned}\frac{\partial^2 \exp\left(-\frac{d_{j,n+k}}{2\sigma^2}\right)}{\partial u_{n+h}\partial u_{n+m}} &= \frac{\partial \exp\left(-\frac{d_{j,n+k}}{2\sigma^2}\right)}{\partial d_{j,n+k}} \frac{\partial^2 d_{j,n+k}}{\partial u_{n+h}\partial u_{n+m}} \\ &+ \frac{\partial^2 \exp\left(-\frac{d_{j,n+k}}{2\sigma^2}\right)}{\partial d_{j,n+k}^2} \frac{\partial d_{j,n+k}}{\partial u_{n+h}} \frac{\partial d_{j,n+k}}{\partial u_{n+m}}\end{aligned} \tag{6.21}$$

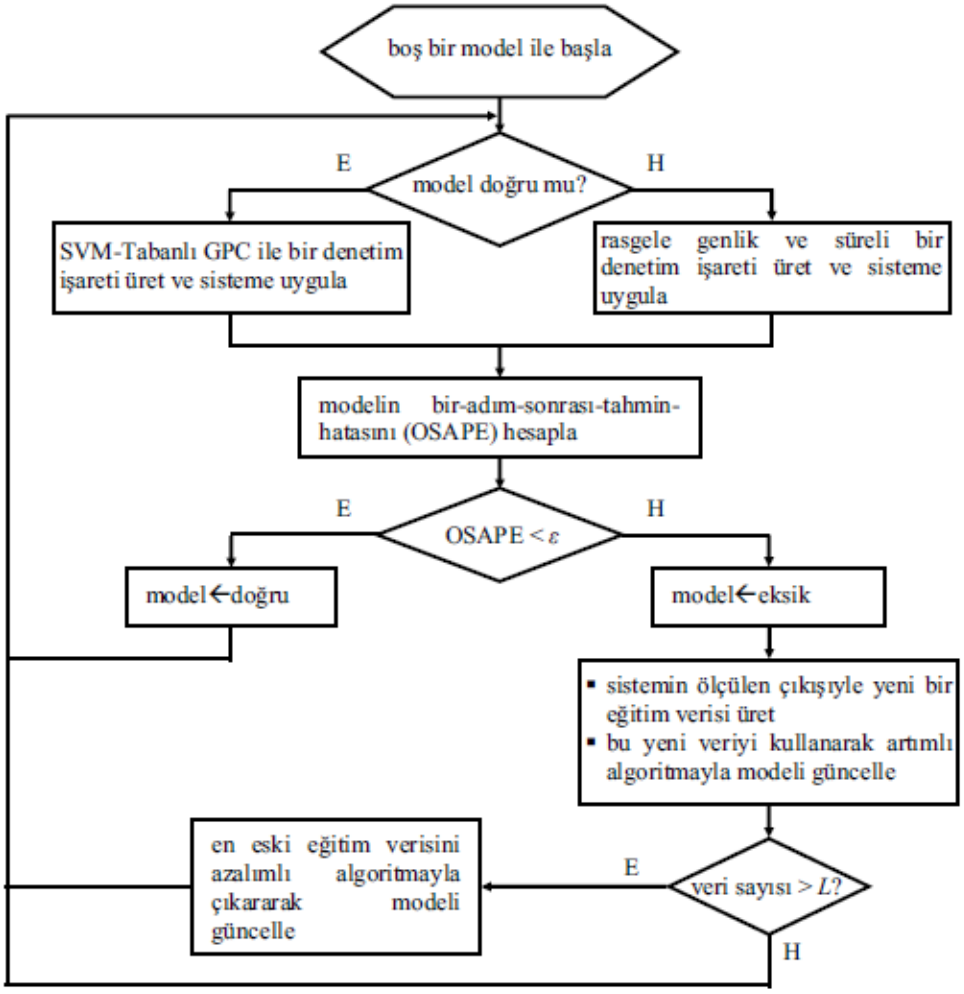
$$\begin{aligned}\frac{\partial^2 d_{j,n+k}}{\partial u_{n+h}\partial u_{n+m}} = -2 &\sum_{i=1}^{\min(k,n_y)} \frac{\partial \hat{y}_{n+k-i}}{\partial u_{n+h}} \frac{\partial \hat{y}_{n+k-i}}{\partial u_{n+m}} \delta_1(k-i-1) \\ -2 &\sum_{i=1}^{\min(k,n_y)} t_{j,n_u+i+1} \frac{\partial^2 \hat{y}_{n+k-i}}{\partial u_{n+h}\partial u_{n+m}} \delta_1(k-i-1) \\ +2 &\sum_{i=1}^{\min(k,n_y)} \hat{y}_{n+k-i} \frac{\partial^2 \hat{y}_{n+k-i}}{\partial u_{n+h}\partial u_{n+m}} \delta_1(k-i-1)\end{aligned}$$

Artık, birinci ve ikinci dereceden terimler eğim vektörü ve Hessian matrisinin hesabında kullanılabilir.

6.3 Çevrimiçi-SVM-Tabanlı MPC Algoritması

Şekil 6-2'deki akış şemasından da görüldüğü gibi Çevrimiçi-SVM-Tabanlı MPC algoritması boş bir model ile başlar (İplikçi 2006[b]). Modelin doğruluğu, modelin bir önceki adımda ürettiği bir-adım-sonrası-tahmin-hatası ($OSAPE = |y_n - \hat{y}_n|$) değerinin ε değerinden küçük olup olmadığına bakılarak karar verilir. Her iterasyonda, eğer model doğru olarak kabul edilmişse o zaman SVM-Tabanlı MPC mekanizmasının ürettiği denetim işareti sisteme uygulanır aksi halde sisteme, genliği $[u_{min}, u_{max}]$ aralığında, süresi de $[\tau_{min}, \tau_{max}]$ aralığında rastgele değişen bir

denetim işareti uygulanır. Sistemin uygulanan işarete olan cevabı ölçüldükten sonra, bir sonraki adımda kullanılmak üzere modelin doğruluğu bu ölçümle belirlenir. Eğer model doğru ise herhangi bir güncelleme yapılmaz. Aksi halde, yani model doğru değilse, sistemin çıkış büyüklüğü ile yeni bir eğitim verisi elde edilir ve artım algoritmasıyla model güncellenir. Eğer eğitim verisi L'yi aşarsa o zaman da azalım algoritmasıyla en eski veri noktası eğitim kümesinden çıkarılır ve böylece algoritmanın mümkün olduğunca en az miktarda hafıza kullanması sağlanır.



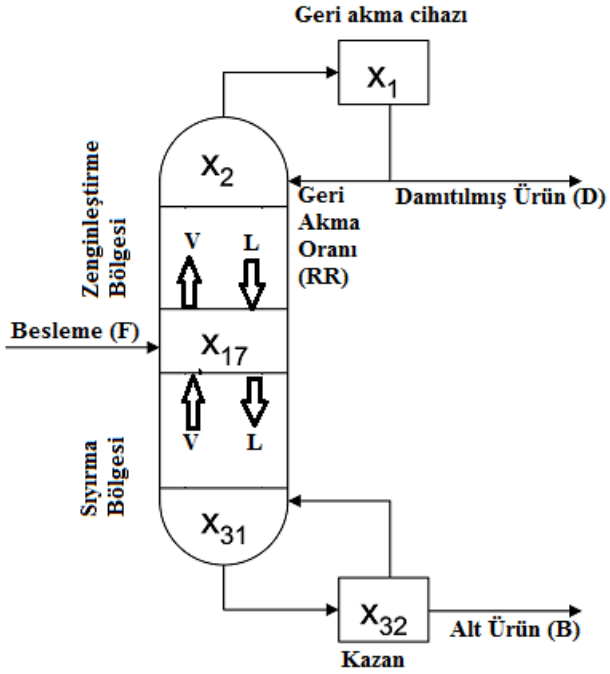
Şekil 6-2: Çevrimiçi-SVM-Tabanlı MPC akış şeması

6.4 Damıtma Kolonu Sisteminin Çevrimiçi Destek Vektör Makineleri Tabanlı Model Öngörülü Denetimi

Damıtma kolonu iki ya da daha fazla sıvı karışımını, buharlaşma noktaları farkından yararlanarak ayırmak için kullanılır. İkili damıtma kolonu, bir girdi karışımına karşılık, çıktı olarak iki sıvıya ayırır. Hafif olan, düşük kaynama noktalı ürün, damıtılmış ürün D olarak adlandırılırken; daha ağır olan alt ürün, B olarak adlandırılır.

Alt ürün kalitesiyle damıtılmış ürün kalitesi arasında ilişki vardır. Birini kontrol etmeye çalışırken diğeri bozulabilir. İkisini birden kontrol etmek zor bir işlem olduğundan genelde ürünlerden biri denetim için seçilir.

D ve B'nin istenen referans noktasına ulaşabilmesi için damıtılmış ürünün kolona geri dönüş oranı RR kontrol edilir. Şekil 6-3'te damıtma kolonu şematik olarak gösterilmiştir.



Şekil 6-3: Damıtma kolonu şematik gösterimi

Damıtma kolonu doğrusal bir sistemdir ve matematiksel modeli aşağıdaki gibidir (Hahn ve Edgar 2002):

$$\frac{dx_{A,1}}{dt} = \frac{1}{A_{Cond}} V(y_{A,2} - x_{A,1})$$

$$\frac{dx_{A,i}}{dt} = \frac{1}{A_{Tray}} \left[L_1(x_{A,i-1} - x_A) - V(y_{A,i} - y_{A,i+1})\right], \qquad i = 2, \dots, 16$$

$$\frac{dx_{A,17}}{dt} = \frac{1}{A_{Tray}} \left[F x_{A,Feed} + L_1 x_{A,16} - L_2 x_{A,17} - V(y_{A,17} - y_{A,18})\right] \tag{6.22}$$

$$\frac{dx_{A,i}}{dt} = \frac{1}{A_{Tray}} \left[L_2(x_{A,i-1} - x_A) - V(y_{A,i} - y_{A,i+1})\right], \qquad i = 18, \dots, 31$$

$$\frac{dx_{A,32}}{dt} = \frac{1}{A_{Reboiler}} \left[L_2 x_{A,31} - (F - D) x_{A,32} - V y_{A,32}\right]$$

Ek denklemler;

$$V = L_1 + D$$

$$L_2 = F + L_1$$

$$RR = \frac{L_1}{D} \tag{6.23}$$

$$\propto_{A,B} = \frac{y_A(1 - x_A)}{(1 - y_A)x_A}$$

Burada;

A_{Cond}: Geri akma cihazında tutulan molar birikim

A_{Tray}: Bölgelerdeki molar birikim

$A_{Reboiler}$: Kazanda tutulan molar birikim

F: Besleme molar akış hızı

D: Damıtılmış ürün molar akış hızı

B: Alt ürün molar akış hızı

L_1: Zenginleştirme bölgesindeki sıvının molar akış hızı

L_2: Sıyırma bölgesindeki sıvının molar akış hızı

V: Buhar molar akış hızı

RR: Geri akma oranı

$\propto_{A,B}$: Uçuculuk

$x_{A,i}$: A bileşiminin i. bölgedeki sıvı mol kesri

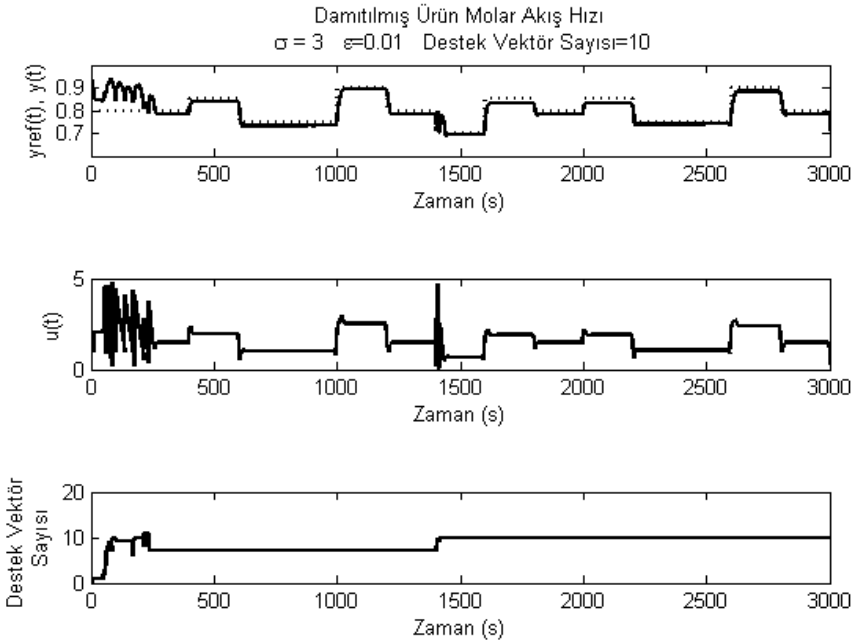
$y_{A,i}$: A bileşiminin i. bölgedeki buhar mol kesridir.

Denetim sürecinin amacı, uygun denetim işaretleriyle ($u(t) = RR$, geri akma oranı) sistem çıkışını ($y(t) = x_1(t)$, damıtılmış ürün molar akış hızı veya $y(t) = x_{32}(t)$, alt ürün molar akış hızı) referans işaretine mümkün olduğunca yakın tutmaktır. Benzetimlerde, denetim işaretinin genliği $u_{min} = 0$ ve $u_{max} = 5$ arasında tutulmuş, denetim işaretinin süresi ise $\tau_{min} = \tau_{max} = 1\,s$ olarak sabit alınmıştır. Başarım göstergesi denklem (6.2)'deki parametrelerden $N_2 = 10$, $N_u = 2$ ve $j = 1, \dots, N_2$ için $\lambda_j = 0.0001$ olarak seçilmiştir. Benzetimlerde sistemin diferansiyel denklemleri 0.1 sn sabit zaman aralığına sahip 4. dereceden Runge Kutta tekniği ile çözülmüştür. MPC algoritmasının uygulandığı her bir örnekleme anında, en iyi arama yönü Değiştirilmiş Newton algoritmasıyla bulunmuştur. Benzetimlerde $\sigma = 3$ ve $C = 1000$ olarak sabit tutulmuştur. Ayrıca çevrimiçi SVM-Tabanlı MPC yönteminin ölçüm gürültülerine karşı dayanıklılığını test etmek için sistemin çıkışına ölçülen büyüklüğe sıfır ortalama değerli Gauss gürültü eklenmiştir. Ölçülen işaretin genliğinin eklenen gürültünün genliğine oranı (SNR) 40 dB olup bu oran denklem (6.24) ile verilmiştir.

$$SNR = 10\log_{10}\left(\frac{\sigma_y^2}{\sigma_v^2}\right) dB \tag{6.24}$$

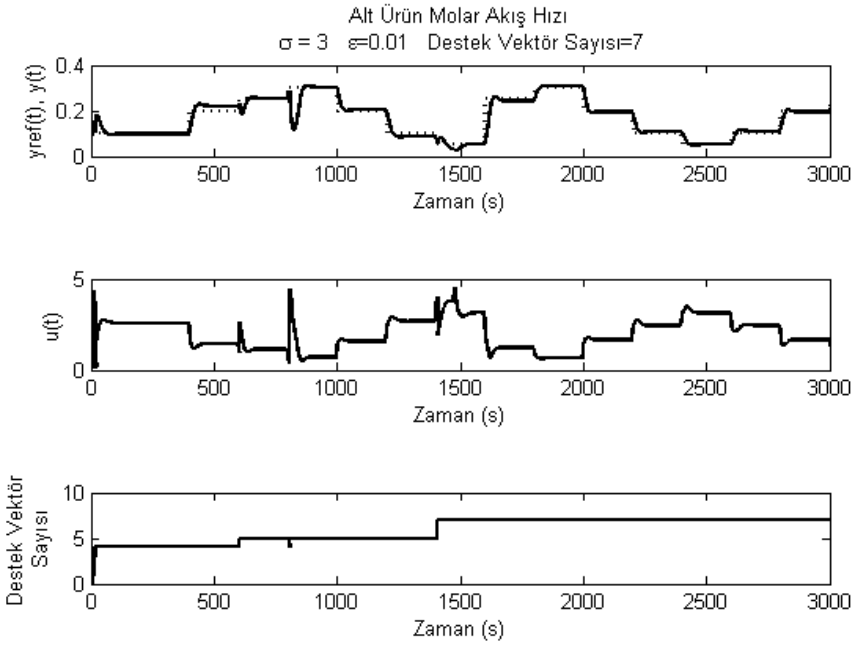
Burada σ_y^2 ve σ_v^2 sırasıyla ölçülen işaretin ve eklenen gürültünün değişinti değerleridir.

Damıtılmış ürün molar akış hızı için basamak referans işareti ve gürültüsüz durum için elde edilen benzetim sonuçları Şekil 6-4'te verilmiştir. Bu benzetimde $\varepsilon = 0.01$ seçilmiştir.



Şekil 6-4: Damıtılmış ürün molar akış hızı için basamak referans işareti ve gürültüsüz durum

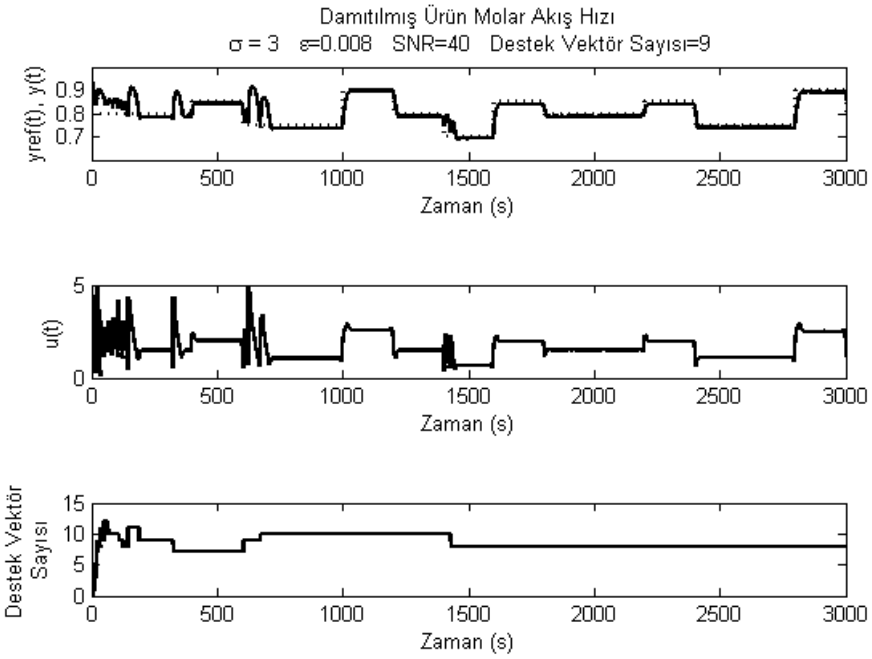
Alt ürün molar akış hızı için basamak referans işareti ve gürültüsüz durum için elde edilen benzetim sonuçları Şekil 6-5'te verilmiştir.



Şekil 6-5: Alt ürün molar akış hızı için basamak referans işareti ve gürültüsüz durum

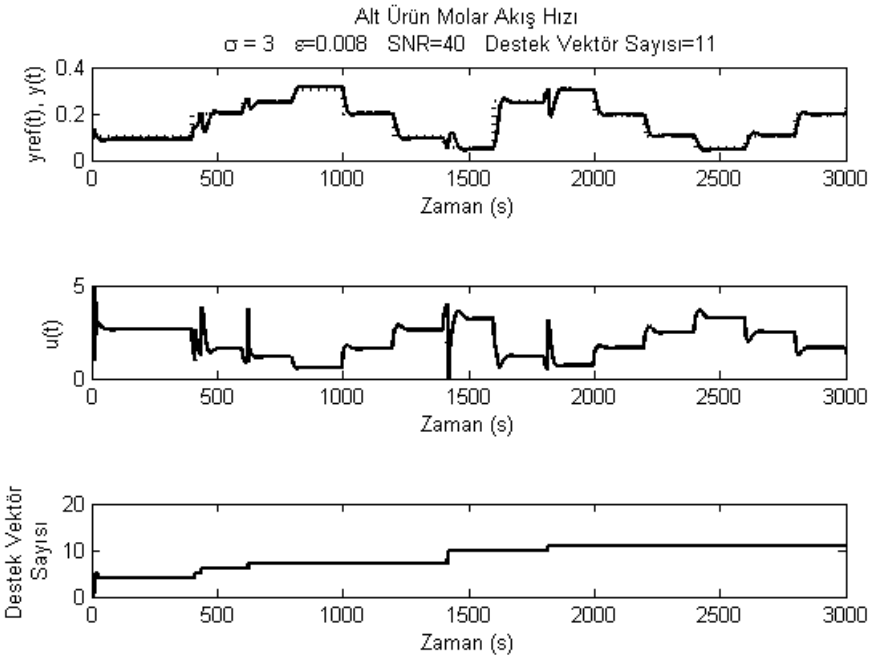
Şekillerden de görüleceği gibi, çevrimiçi SVM-Tabanlı MPC denetim mekanizmasında başlangıçta boş olan SVM modeli Şekil 6-2'deki akış şemasındaki algoritmaya göre güncellenmektedir. Model, her güncellemede sistemin dinamiğini daha iyi temsil etmekte ve böylece sistem çıkışı, referans işaretini daha iyi takip edebilmektedir. Belli bir zamandan sonra artık güncelleme gerekliliği iyice azalmakta ve hatta sistem parametrelerinde veya çevresel koşullarda bir değişim olmadığı sürece de hiç güncelleme yapılmamaktadır.

Damıtılmış ürün molar akış hızı için basamak referans işareti ve 40 dB gürültülü durum için elde edilen benzetim sonuçları Şekil 6-6'da verilmiştir. Bu benzetimde $\varepsilon = 0.008$ seçilmiştir.



Şekil 6-6: Damıtılmış ürün molar akış hızı için basamak referans işareti ve 40 dB gürültülü durum

Alt ürün molar akış hızı için basamak referans işareti ve 40 dB gürültülü durum için elde edilen benzetim sonuçları Şekil 6-7'de verilmiştir. Bu benzetimde $\varepsilon = 0.008$ seçilmiştir.

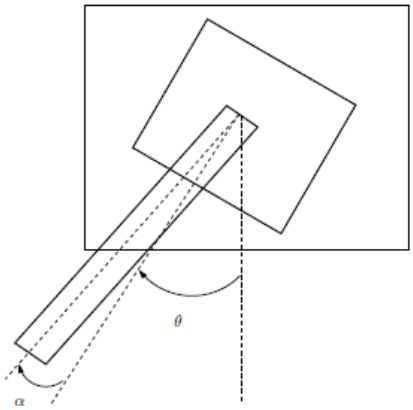


Şekil 6-7: Alt ürün molar akış hızı için basamak referans işareti ve 40 dB gürültülü durum

Gürültülü durumda da benzer gözlemleri yapabiliriz. Çevrimiçi SVM-Tabanlı MPC denetim mekanizması zamanla sistemin modelini daha iyi temsil etmekte ve denetim işini daha iyi yapmaktadır. Benzetimlere uzun süre devam edildiğinde sistem çıkışının referans işaretini yüksek bir doğrulukla takip etmeye devam ettiği gözlemlenmiştir.

6.5 Esnek Eklemli Tek Uzuvlu Manipülatör Sistemin Çevrimiçi Destek Vektör Makineleri Tabanlı Model Öngörülü Denetimi

Esnek eklemli tek uzuvlu manipülatör sistemi Şekil 6-8'de şematik olarak gösterilmektedir. Manipülatörün dikeyle yaptığı açı θ ile, yayın sapması ise α ile gösterilmiştir (Akyuz ve diğ. 2011).



Şekil 6-8: Esnek eklemli tek uzuvlu manipülatör sistemi

Esnek eklemli tek uzuvlu manipülatör sistemi doğrusal olmayan bir sistem olup matematiksel modeli aşağıdaki gibidir (Akyuz ve diğ. 2011).

$$[x_1\ x_2\ x_3\ x_4]^T = \left[\theta\ \alpha\ \dot{\theta}\ \dot{\alpha}\right]^T$$

$$\dot{x}_1 = x_3$$

$$\dot{x}_2 = x_4$$

$$\dot{x}_3 = \frac{K_s}{J_h}x_2 - \frac{K_m^2 K_g^2}{R_m J_h}x_3 + \frac{K_m K_g}{R_m J_h}u \tag{6.25}$$

$$\dot{x}_4 = -\frac{K_s}{J_h}x_2 + \frac{K_m^2 K_g^2}{R_m J_h}x_3 - \frac{K_m K_g}{R_m J_h}u - \frac{K_s}{J_l}x_2 + \frac{mgh}{J_l}\sin(x_1 + x_2)$$

$$y = x_1 + x_2$$

Burada;

K_s: Yay katsayısı (1.61 $[N/m]$)

J_h: Merkez ataleti (0.0021 $[Kgm^2]$)

m: Yük ağırlığı (0.403 $[Kg]$)

g: Yerçekimi (-9.81 $[N/m]$)

h: Yükseklik (0.06 $[m]$)

K_m: Motor sabiti $\left(0.00767\ \left[\frac{N}{rad/s}\right]\right)$

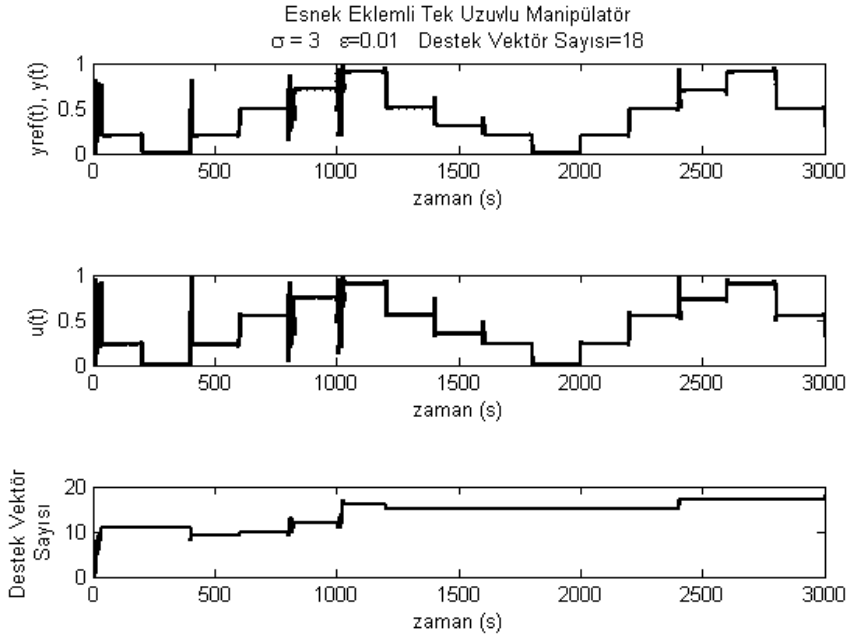
K_g: Dişli oranı (70)

J_l: Yük ataleti ($0.0059\ [Kgm^2]$)

R_m: Motor direnci ($2.6\ [\Omega]$)

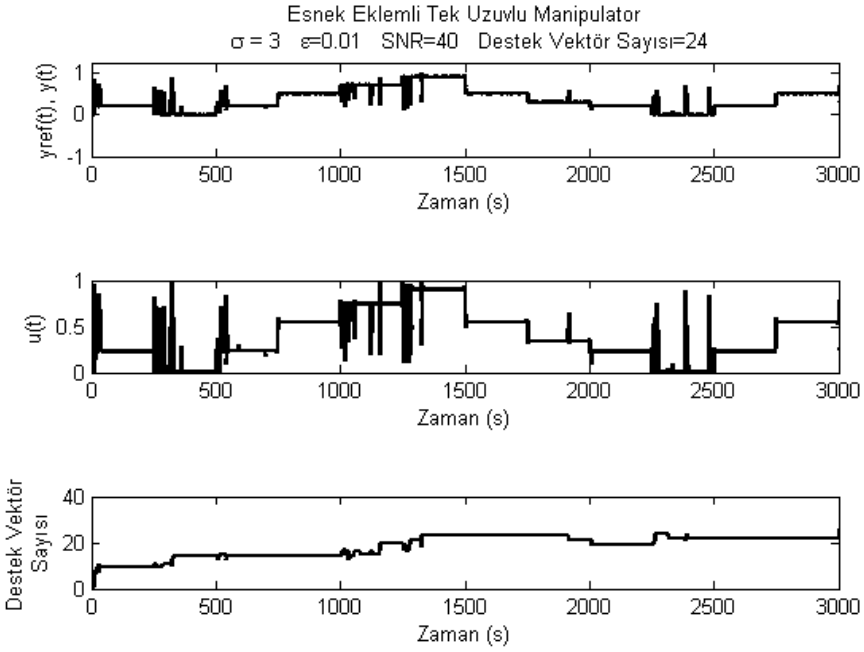
Benzetimlerde, denetim işaretinin genliği $u_{min} = 0$ ve $u_{max} = 1$ arasında tutulmuş, denetim işaretinin süresi ise $\tau_{min} = \tau_{max} = 1\ s$ olarak sabit alınmıştır. Başarım göstergesi denklem (6.2)'deki parametrelerden $N_2 = 10$, $N_u = 2$ ve $j = 1, \dots, N_2$ için $\lambda_j = 0.0001$ olarak seçilmiştir. Benzetimlerde sistemin diferansiyel denklemleri 10^{-3} sn sabit zaman aralığına sahip 4. dereceden Runge Kutta tekniği ile çözülmüştür. MPC algoritmasının uygulandığı her bir örnekleme anında, en iyi arama yönü Değiştirilmiş Newton algoritmasıyla bulunmuştur. Benzetimlerde $\sigma = 3$ ve $C = 1000$ olarak sabit tutulmuştur. Ayrıca çevrimiçi SVM-Tabanlı MPC yönteminin ölçüm gürültülerine karşı dayanıklılığını test etmek için sistemin çıkışına ölçülen büyüklüğe sıfır ortalama değerli Gauss gürültü eklenmiştir. Ölçülen işaretin genliğinin eklenen gürültünün genliğine oranı (SNR) 40 dB'dir.

Gürültüsüz durum için elde edilen benzetim sonuçları Şekil 6-9'da verilmiştir. Bu benzetimde $\varepsilon = 0.01$ seçilmiştir.



Şekil 6-9: Esnek eklemli tek uzuvlu manipülatör sistemi için gürültüsüz durum için elde edilen benzetim sonuçları

Basamak referans işareti ve 40 dB gürültülü durum için elde edilen benzetim sonuçları Şekil 6-10'da verilmiştir. Bu benzetimde $\varepsilon = 0.01$ seçilmiştir.



Şekil 6-10: Esnek eklemli tek uzuvlu manipülatör sistemi için gürültülü durum için elde edilen benzetim sonuçları

7. SONUÇLAR

Çevrimiçi-SVM-Tabanlı MPC yöntemiyle hem doğrusal sistemlerin hem de doğrusal olmayan sistemlerin çevrimiçi denetimi gerçekleştirilmiştir. Bu yöntemde, matematiksel modeli bilinmeyen sistemin dinamik davranışını temsil edecek olan ve başlangıçta boş olan SVM modelinin, çevrimiçi destek vektör bağlanımı algoritmasıyla denetim esnasında gerektiğinde güncellenerek MPC döngüsü içinde kullanılmasıyla modelleme ve denetim eşzamanlı olarak gerçekleştirilmiştir. Benzetim sonuçları göstermiştir ki, çevrimiçi-SVM-Tabanlı MPC yapısı basamak tipli referans işareti için oldukça iyi bir denetim başarımı sağlamaktadır. Başka bir deyişle, matematiksel modeli bilinmeyen bir sistem çevrimiçi-SVM-Tabanlı MPC ile kontrol edildiğinde önceden bir referans işaretini çok küçük geçici ve sürekli hal hatalarıyla takip edebilmektedir. Sonuç olarak, çevrimiçi-SVM-Tabanlı MPC yöntemi denetim işini oldukça iyi bir başarımla gerçekleştirmiştir ve diğer çevrimiçi yaklaşımlara alternatif olarak kullanılabilir. Bunun yanısıra, bu yöntemin, SVM tekniklerindeki gelişmelere paralel olarak daha da gelişmesi mümkündür.

8. KAYNAKLAR

Akyuz, I. H., Yolacan, E., Ertunc, H. M., and Bıngul, Z., "PID and State Feedback Control of a Single Flexible Joint Robot Manipulator", *2011 IEEE International Conference on Mechatronics*, 409-414, (2011).

Cauwenberghs, G. and Poggio, T., "Incremental and Decremental Support Vector Machine Learning", (eds: Leen, T. K., Dietterich, T. G. and Tresp, V.), *Advanced in Neural Information Processing Systems*, 13, Cambridge, MA: MIT Press., (2001).

Clarke, D. W., Mohtadi, C. and Tuffs, P. C., "Generalized Predictive Control-Part 1: The Basic Algorithm", *Automatica*, 23, 137-148, (1987).

Clarke, D. W. and Mohtadi, C., "Properties of Generalized Predictive Control", *Automatica*, 25, 859-875, (1989).

Cristianini, N and Taylor, J. S., *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*, New York: Cambridge University Press, (2000).

Hahn, J. and Edgar, T. F., "An Improved Method For Nonlinear Model Reduction Using Balancing Of Empirical Gramians", *Computers and Chemical Engineering*, 26, 1379-1397, (2002).

İplikci, S., "Support Vector Machines Based Generalized Predictive Control", *Int'l. J. of Ad. Cont. And Sig. Proc.*, 16, 843-862, (2006).

İplikci, S., "Online Trained Support Vector Machines Based Generalized Predictive Control of Nonlinear Systems", *Int'l. J. of Ad. Cont. And Sig. Proc*., 20, 599-621, (2006).

İplikci, S., *Optimizasyon Teknikleri Ders Notları*, Pamukkale Üniversitesi Elektrik-Elektronik Mühendisliği Bölümü, Denizli, (2013).

Ma, J., Theiler, J. and Perkins, S., "Accurate Online Support Vector Regression", *Neural Computation*, 15, 2683-2703, (2003).

Martinez, M., Senent, J. S. and Blasco, X., "Generalized Predictive Control Using Genetic Algorithms (GAGPC)", *Engineering Applications of Artificial Intelligence*, 11, 355-367, (1998).

Nocedal, J. and Wright, S. J., *Numerical optimization*, New York*:* Springer-Verlag, (1999).

Parrella, F., "Online Support Vector Regression", Ph.D Thesis, Department of Information Science University of Genoa, Genoa, (2007).

Qin, S. J. and Badgwell, T. A., "A Survey of Industrial Model Predictive Control Technology", *Control Engineering Practice*, 11, 733-764, (2003).

Richalet, J. A., Rault, A., Testud, J. L. and Papon, J., "Model Predictive Heuristic Control: Applications To An Industrial Process", *Automatica*, 14, 413-428, (1978).

Richalet, J., "Industrial Applications of Model-Based Predictive Control", *Automatica*, 29, 1251-1274, (1993).

Saunders, C., Gammerman, A. and Vovk, V., "Ridge Regression Learning Algorithm in Dual Variables", *Proceedings of the 15th Int'l. Conference on Machine Learning ICML'98*, 515-521, (1998).

Vapnik, V., *The Nature of Statistical Learning Theory*, London: Springer-Verlag, (1995).

Vapnik, V., *Statistical Learning Theory*, New York: John Willey, (1998).

Vapnik, V., "The Support Vector Method of Function Estimation", (eds: Suykens J. A. K, Vanderwalle, J.), *Nonlinear Modeling Advanced Black Box Techniques*, Boston: Kluwer Academic Publishers, 55-85, (1998).

Yang, W. Y., Cao, W., Chung T. S. and Morris J., *Applied numerical methods using Matlab*, New Jersey: Wiley Interscience, (2005).

9. ÖZGEÇMİŞ

Ad Soyad : Merve TOPALOĞLU

Doğum Yeri ve Tarihi : Denizli – 19.03.1989

Lisans Üniversite : Pamukkale Üniversitesi

Elektronik posta : mervetopaloglu@windowslive.com

İletişim Adresi : Yeşilköy Mahallesi 529 Sokak Yaşamkent Sitesi NO:11-A D:8 Denizli