YALOVA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

-BİTİRME TEZİ-

YAPAY SİNİR AĞLARI İLE TENSORFLOW KÜTÜPHANESİNİ KULLANARAK HAVA DURUMU TAHMİNİ

Burak SERTTAŞ
140101058

1. Bitirme Tezi Danışmanı : Dr.Öğr.Üyesi Osman Hilmi KOÇAL
2. Jüri Üyesi : Dr.Öğr.Üyesi Adem TUNCER
3. Jüri Üyesi : Öğr.Gör.Dr. Yunus ÖZEN

YALOVA, 2019

YALOVA ÜNİVERSİTESİ
MÜHENDİSLİK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

YAPAY SİNİR AĞLARI İLE TENSORFLOW KÜTÜPHANESİNİ KULLANARAK HAVA DURUMU TAHMİNİ

Burak SERTTAŞ
140101058

1. Bitirme Tezi Danışmanı : Dr.Öğr.Üyesi Osman Hilmi KOÇAL
2. Jüri Üyesi : Dr.Öğr.Üyesi Adem TUNCER
3. Jüri Üyesi : Öğr.Gör.Dr. Yunus ÖZEN

Bitirme Tezinin Dönemi: 2018 – 2019 Güz Yarıyılı

İÇİNDEKİLER

İÇİNDEKİLER iii

KISALTMA LİSTESİ iv

ŞEKİL LİSTESİ v

ÖNSÖZ vi

ÖZET vii

ABSTRACT viii

1. GİRİŞ 1

2. MATERYAL ve YÖNTEMLER 2

2.1 Çok katmanlı algılayıcılar 2
2.2 Dereceli azalma çeşitleri ve optimizasyon algoritmaları 3
2.3 Adam optimizasyon algoritması 6
2.3.1 Adam algoritmasının çalışma mantığı 7
2.3.2 Adam yapılandırma parametreleri 9
2.4 XOR problemi 10
2.5 Projede kullanılan ortam (IDE) ve kütüphaneler 10
2.5.1 Kullanılan kütüphaneler 11
2.5.1.1 Tensorflow 11
2.5.1.2 Matplotlib 11
2.5.1.3 CSV 11
2.5.2 Kullanılan ortam (IDE) ve programlama dili 12
2.6 Min-Max normalizasyonu 14

3. UYGULAMA ve DENEYSEL SONUÇLAR 15

3.1 Veri seti 15
3.2 YSA modelinin yapısı ve parametreleri 15
3.3 Deneysel sonuçlar 17

4. SONUÇ ve ÖNERİLER 20

KAYNAKLAR 21

KISALTMA LİSTESİ

ADAGRAD	Adaptive Gradient Algorithm
ADAM	Adaptive Moment Estimation
CSV	Comma Separated Values
ÇKA	Çok Katmanlı Algılayıcılar
IDE	Integrated Development Enviroment
MLP	Multilayer Perceptron
MAX	Maksimum
MİN	Minimum
NLP	Natural Language Processing
NORM	Normalizasyon
RMSPROP	Root Mean Square Propagation
TF	Tensorflow
YSA	Yapay Sinir Ağları
XOR	eXclusive OR

ŞEKİL LİSTESİ

Şekil 2.1 ÇKA modeli 2
Şekil 2.2 Batch gradient descent 4
Şekil 2.3 Stochastic gradient descent 4
Şekil 2.4 Mini-Batch gradient descent 5
Şekil 2.5 Öğrenme oranlarındaki değişimler 6
Şekil 2.6 Adam algoritmasının learning_rate değerlerinin karşılaştırılması 7
Şekil 2.7 Optimizasyon algoritmalarının kayıp değerlerinin karşılaştırılması 8
Şekil 2.8 Projede kullanılan adam algoritmasının parametreleri 9
Şekil 2.9 Basit XOR problemi 10
Şekil 2.10 Sınıflandırılamayan XOR problemi 10
Şekil 2.11 Günümüzde veri bilimi için en çok kullanılan platformlar 12
Şekil 3.1 Tasarlanan sistem için YSA blok diyagramından bir kısım 17
Şekil 3.2 Nem değerleri için 10000 epoch sonucundaki hata oranları 18
Şekil 3.3 Gerçek ve tahmin edilen nem değerleri 18
Şekil 3.4 Sıcaklık değerleri için 10000 epoch sonucundaki hata oranları 19
Şekil 3.5 Gerçek ve tahmin edilen sıcaklık değerleri 19

ÖNSÖZ

‘Yapay Sinir Ağları İle Tensorflow Kütüphanesini Kullanarak Hava Durumu Tahmini’ konulu çalışma, Yalova Üniversitesi, Bilgisayar Mühendisliği Bölümünde “Bitirme Tezi” olarak hazırlanmıştır.

Bitirme tezi danışmanlığımı üstlenen, beni yönlendiren ve her konuda bana her türlü desteği veren saygıdeğer hocam Dr. Öğretim Üyesi Osman Hilmi KOÇAL’a, bana her zaman güvenen ve destek olan aileme ve okul hayatım boyunca bana destek olan tüm hocalarıma ve arkadaşlarıma teşekkür eder, saygılarımı sunarım.

Burak SERTTAŞ

ÖZET

Yapay sinir ağları, sınıflandırma, modelleme ve tahmin gibi birçok günlük hayat probleminin çözümünde başarılı sonuç veren bir yöntemdir. Yapay sinir ağları nöronlar arasındaki bağlantı ağırlıklarını ayarlayarak öğrenme gerçekleştirir. Bu çalışmada, yapay sinir ağları yöntemiyle çok katmanlı algılayıcı modeli kullanılarak XOR problemi çözülmeye çalışılmıştır. Bu problemin çözümünde tensorflow kütüphanesinden yararlanılmıştır. Bilindiği gibi tek katmanlı algılayıcılar lineer problemlere çözüm üretebilirken lineer olmayan problemlere çözüm üretememiş, lineer olmayan problemler yapay sinir ağları ile çok katmanlı algılayıcı modelinin geliştirilmesi ile çözülebilmişlerdir. Tek katmanlı algılayıcılarda mantıksal AND ve OR problemleri doğrusal olarak sınıflandırılabilirken, XOR probleminin tek bir doğru ile sınıflandırılamadığı gözlemlenebilmektedir. Çok katmanlı algılayıcılarda ise, yapay sinir ağlarında yaygın olarak kullanılan geri yayılım algoritması ile tahmin problemleri çözülebilmektedir. Bu projede, optimizasyonu sağlamak için dereceli azalma optimizasyon algoritmalarından adam algoritması kullanılmıştır. Ağın katman sayısı, katmanlardaki nöron sayısı, epoch sayısı, öğrenme oranı, momentum katsayısı, aktivasyon fonksiyonu, normalizasyon yöntemi, başlangıç ağırlıkları gibi parametrelerin değiştirilerek ağın eğitilmesi sağlanabilmekte ve eğitilen ağ test edilerek ağın performans ölçümü yapılabilmektedir. Bu çalışmada da, tensorflow kütüphanesi yardımıyla bazı parametreler kullanılarak ağ eğitilip, daha sonra XOR problemini çözebilecek duruma getirilip, hava durumu tahmini yapılmak istenmiştir. Bunun için uygun veri seti hazırlandıktan sonra görselleştirilip, bir sonraki günün hava sıcaklığı tahmin edilmeye çalışılmıştır.

Anahtar sözcükler: Yapay sinir ağları, Dereceli azalma algoritması, XOR problemi, Tensorflow kütüphanesi, Python, Tahmin

ABSTRACT

Artificial neural networks are a succesful method for solving many daily life problems such as classification, modeling and estimation. Artificial neural networks perform learning by adjusting the connection weights between the units. In this study, XOR problem has been tried to be solved by using multi-layer perceptron model with artificial neural network method. To solve this problem, tensorflow library was used. As it is known, single layer perceptron model can produce solutions to linear problems, but can not nonlinear problems, they solved by multi-layer perceptron models with artificial neural networks. While the logical AND and OR problems can be classified linearly in single layer perceptron, it can be observed that the XOR problem cannot be classified by a single line. In multi-layer perceptron, estimation problems can be solved with back propagation algorithm which is widely used in artificial neural networks. In this project, man algorithm is used from gradient descent optimization algorithms to ensure optimization. The network can be trained by changing the parameters such as the number of layers, the number of epoch, learning rate, momentum constant, activation function, normalization method, initial weights, and network performance can be measured by testing the trained network. In this study, the network was used by using some parameters with the help of tensorflow library to solve the XOR problem. For this purpose, after the prepation of the appropriate dataset, visualization of the next day's air temperature has been estimated.

Keywords: Artificial neural networks, Gradient descent algorithm, XOR problem, Tensorflow library, Python, Prediction

1. GİRİŞ

Günlük hayatımızda vazgeçilmez bir yere sahip olan bilgisayarın, artık insanlar gibi karar verme ve öğrenme gerçekleştirme yeteneklerini kazanması ile kullanım alanları oldukça genişlemiştir. Matematiksel olarak ifade edilemeyen ve insanlar tarafından çözülmesi zor olan problemlerin çözümünde yapay zekâ yöntemleri kullanılmaktadır. Yapay zekâ yöntemlerinin en temel özelliği, olaylara ve problemlere çözümler üretirken örneklerden, tecrübeden, benzetmelerden öğrenme gerçekleştirme ve karar verebilme yeteneklerinin olmasıdır.

Makine öğrenimi için en popüler yaklaşımlardan biri yapay sinir ağlarıdır. Yapay sinir ağları, geleneksel hesaplama yöntemleri ile çözülemeyen problemlerin çözümünde yaygın olarak kullanılmaktadır.

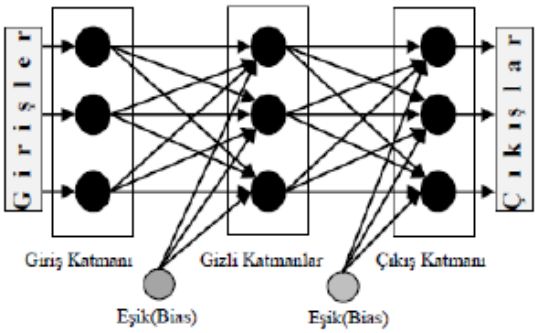
Öğrenme, genelleme, doğrusal olmama, hata toleransı, uyum, paralellik gibi üstünlüklere sahip olan yapay sinir ağları; görüntü ve sinyal işleme, hastalık tahmini gibi tıbbi uygulamalarda, mühendislik, üretim, finans, optimizasyon, sınıflandırma gibi çok farklı uygulama alanlarında kullanılmaktadır.

Yapay sinir ağları ile çeşitli programlama dillerinde birçok kütüphane ve araç geliştirilmiştir. Bu çalışmada, Python programlama dili kullanılarak YSA'nın öğrenme, genelleme, özelliklerinin kolayca görülebileceği, tahmin probleminin uygulanabileceği görsel ve esnek bir yazılım geliştirilmiştir. Tensorflow kütüphanesinden de yararlanılmıştır. Yazılımda çok katmanlı algılayıcı gradient descent algoritma yöntemlerinden olan adam algoritması kullanılarak tasarlanmıştır. Böylelikle hata en aza indirilmeye çalışılmıştır. Yazılımın çok katmanlı algılayıcı bölümünde, ağın yapısı, katman ve katmanlardaki nöron sayısı, verilerin normalizasyon yöntemi, katmanlarda kullanılan aktivasyon fonksiyonu, öğrenme oranı, momentum katsayısı, epoch sayısı, eğitimi sonlandırma koşulu, başlangıç ağırlıklarının seçimi tensorflow kütüphanesinden yararlanılarak oluşturulmuştur. Eğitim esnasında oluşan bazı kayıp değerler gözlemlenebilmektedir. Test işleminden sonra ağın performansı ölçülebilmektedir. Burada XOR problemini çözebilen ve gelecekteki hava durumunu bu probleme uygun olarak tahmin edebilen ve bunun için yapay sinir ağları yönteminden faydalanan bir yazılım amaçlanmaktadır.

2 MATERYAL ve YÖNTEMLER

2.1 Çok Katmanlı Algılayıcılar

Çok katmanlı algılayıcıların doğrusal olmayan problemlerin çözümünde başarısız olmasının üzerine geliştirilen çok katmanlı algılayıcılar (ÇKA), bilgi girişinin yapıldığı girdi katmanı, bir veya daha fazla gizli (ara katman) ve bir çıktı katmanından oluşmaktadır. ÇKA'da katmanlar arası ileri ve geri yayılım olarak adlandırılan geçişler bulunur. İleri yayılım safhasında, ağın çıktısı ve hata değeri hesaplanır. Geri yayılım safhasında ise hesaplanan hata değerinin minimize edilmesi için katmanlar arası bağlantı ağırlık değerleri güncellenir.
Şekil 2.1'de ÇKA yapısı gösterilmiştir. Mimarinin görünümü Şekil 2.1'deki gibidir:



Şekil 2.1 ÇKA modeli [1]

Giriş Katmanı: Yapay sinir ağına dış dünyadan girdilerin geldiği katmandır. Bu katmanda dış dünyadan gelecek giriş sayısı kadar nöron bulunmasına rağmen genelde girdiler herhangi bir işleme uğramadan alt katmanlara iletilmektedir.

Gizli Katmanlar: Giriş katmanından çıkan bilgiler bu katmana gelir. Ara katman sayısı ağdan ağa değişebilir. Bazı yapay sinir ağlarında ara katman bulunmadığı gibi bazı yapay sinir ağlarında ise birden fazla ara katman bulunmaktadır. Ara katmanlardaki nöron sayıları giriş ve çıkış sayısından bağımsızdır. Birden fazla ara katman olan ağlarda ara katmanların kendi aralarındaki nöron sayıları da farklı olabilir. Ara katmanların ve bu katmanlardaki nöronların sayısının artması hesaplama karmaşıklığını ve süresini arttırmasına rağmen yapay sinir ağının daha karmaşık problemlerin çözümünde de kullanılabilmesini sağlar.

Çıkış Katmanı: Ara katmanlardan gelen bilgileri işleyerek ağın girdi katmanından gelen verilere karşılık olan çıktıları üreten katmandır. Bu katmanda üretilen çıktılar dış dünyaya

gönderilir. Geri beslemeli ağlarda bu katmanda üretilen çıktı kullanılarak ağın yeni ağırlık değerleri hesaplanır.

2.2 Dereceli Azalma Çeşitleri ve Optimizasyon Algoritmaları

Dereceli Azalma, optimizasyonu gerçekleştiren en popüler algoritmalardan biridir. Dereceli Azalma (Gradient Descent) üzerinde bu kadar durulmasının sebebi sadece doğrusal regresyon problemlerinde değil başta sinir ağları olmak üzere birçok makine öğrenmesi probleminde kullanılmasıdır. Aynı zamanda birçok derin öğrenme kütüphanesi (lasagne, caffe ve keras), dereceli azalmayı optimize etmek için çeşitli algoritmaların uygulamalarını içerir. Optimizasyonun amacı, önceden tanımlanmış bir zaman içinde mümkün olan en iyi f(x) değerini bulmaktır.

Dereceli Azalma, bir modelin parametreleri (beta/teta) tarafından parametrelendirilen bir hedef fonksiyonunu (maliyet fonksiyonu) en aza indirmenin bir yoludur. Başka bir deyişle, bir vadiye varana kadar yokuş aşağı nesnel fonksiyon tarafından oluşturulan yüzeyin eğim yönünü takip etmektir.

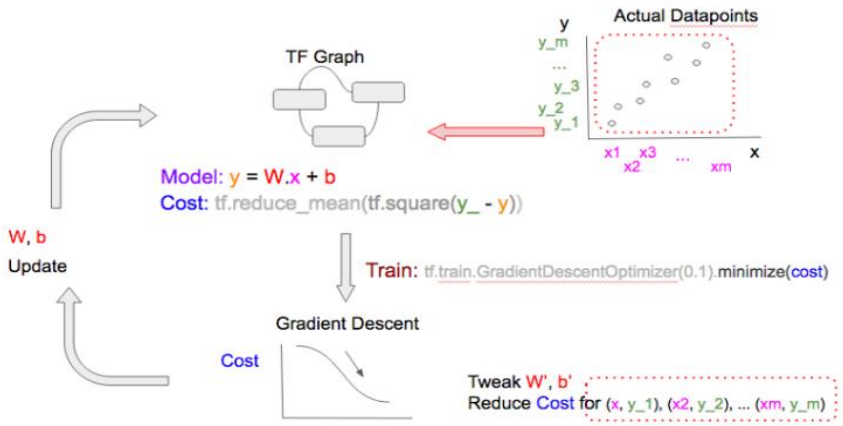
Dereceli Azalmanın üç farklı çeşidi vardır; Bunlar, hedef fonksiyonun gradyanını (derecesini) hesaplamak için ne kadar veri kullandığımız konusunda farklılık gösterir. Veri miktarına bağlı olarak, parametre güncellemesinin doğruluğu ile bir güncelleme gerçekleştirmek için gereken süre arasında bir denge kurarız.

Toplu Dereceli Azalma (Batch Gradient Descent): Vanilya Dereceli Azalması olarak da bilinir. Maliyet fonksiyonunun dereceli azalmasını yani gradyan inişini tüm eğitim verisi seti için hesaplar. Sadece bir güncelleme gerçekleştirmek için tüm veri kümesinin dereceli azalmasını hesaplamamız gerektiğinden, toplu eğim inişi çok yavaş olabilir ve belleğe sığmayan veri kümeleri için de zor olabilir. Buna göre parametre güncellemesi aşağıdaki şekilde gösterilebilir:

$$\theta_t = \theta_{t-1} - \alpha . \nabla_\theta J(\theta) \quad (1)$$

Burada teta güncellenecek parametre vektörü, alfa öğrenme katsayısı ve çıkarmadan sonraki işlem maliyet fonksiyonunun gradientini göstermektedir. Batch Gradient Descent yöntemine göre tek bir güncelleme yapabilmek için tüm veri seti kümesinin gradientini hesaplamak

gerekmektedir. Bundan dolayı bu yöntem çok yavaş çalışır ve büyük miktarda veri setlerinin uygulanmasında bellek problemleri ile karşılaşılabilir. Görünüm Şekil 2.2'de verilmiştir:

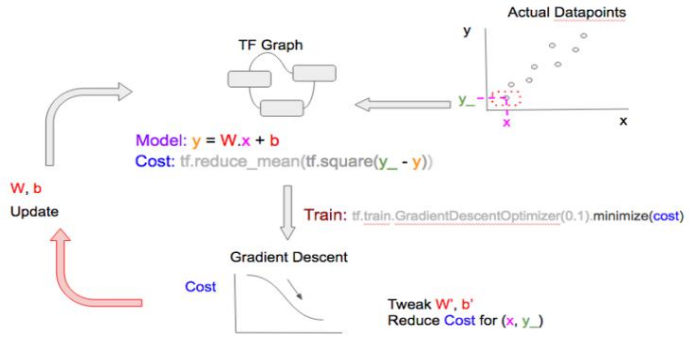


Şekil 2.2 Batch gradient descent [2]

Olasılıksal Dereceli Azalma (Stochastic Gradient Descent): Stochastic Gradient Descent, Batch Gradient Descent'in aksine her eğitim örneği x(i) ve etiket y(i) için bir parametre güncellemesi gerçekleştirir. Olasılıksal Dereceli Azalma, bir kerede bir güncelleme gerçekleştirir. Bu nedenle genellikle daha hızlı çalışır ve yerel minimuma daha erken sürede ulaşılabilir.

$$\theta_t = \theta_{t-1} - \alpha . \nabla_\theta J(\theta; x^i, y^i) \tag{2}$$

Bu çalışmada, Stochastic Gradient Descent yöntemi ile adam optimizasyon algoritması kullanılmıştır. Görünüm Şekil 2.3'te verilmiştir:

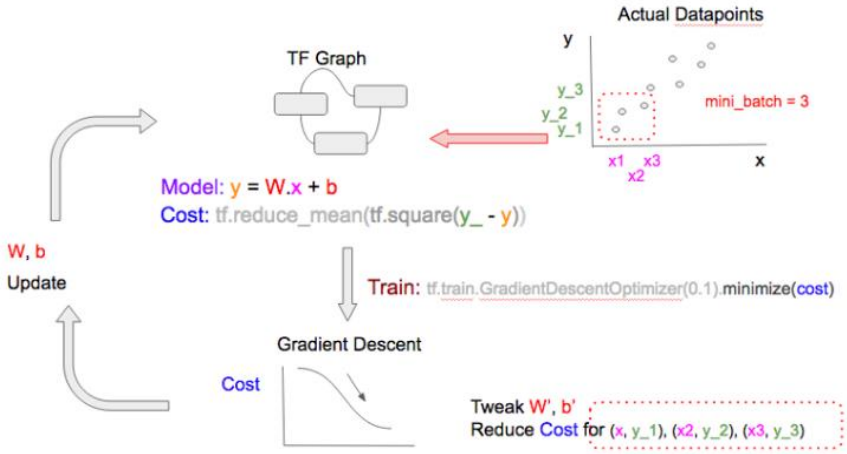


Şekil 2.3 Stochastic gradient descent [2]

Mini- Toplu Dereceli Azalma (Mini-Batch Gradient Descent): Mini Batch Gradient Descent, diğer dereceli azalma algoritmalarının en iyisini alıyor. Her iki yöntemin avantajlarını

barındıracak şekilde düşünülerek geliştirilmiştir. Güncelleme işlemi, her bir n adet eğitim seti için gerçekleştirilir. Görünüm Şekil 2.4'te verilmiştir:

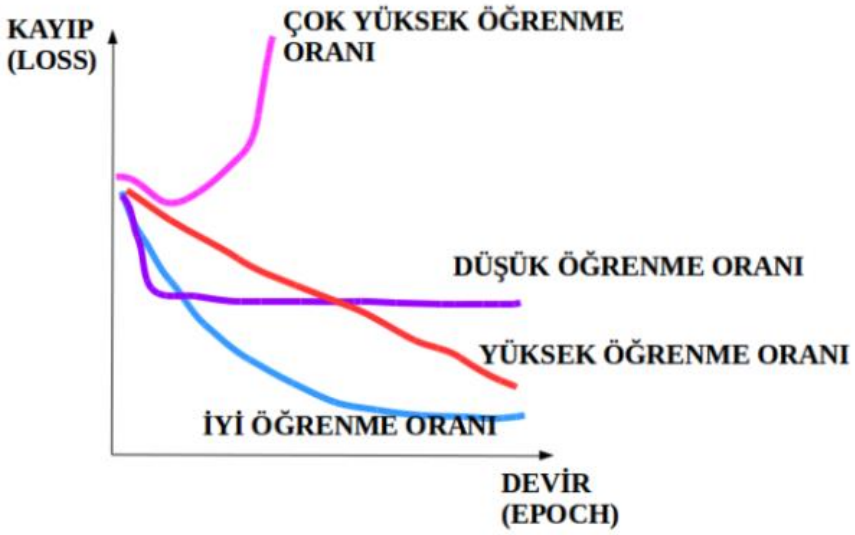
$$\theta_t = \theta_{t-1} - \alpha . \nabla_\theta J(\theta; x^{i+n}, y^{i+n}) \quad (3)$$



Şekil 2.4 Mini-Batch gradient descent [2]

Dereceli Azalma yöntemlerinin sebep olduğu birçok sorun olabilir. Doğru yakınsama için öğrenme oranını seçmek zor olabilir. Bütün ağırlıklar aynı anda güncellenmeyebilir. Bu problemlere çözüm olarak birçok dereceli azalma optimizasyon algoritması kullanılabilir. Bu çalışmada, adam optimizasyon algoritması kullanılmıştır. Bu algoritma stokastik optimizasyon yöntemidir. Bu algoritmadan bahsetmeden önce şuna bakacak olursak; Dereceli Azalma Algoritmasının doğru çalıştığını anlamak için devir sayısı ile maliyet fonksiyonu arasındaki grafiğe bakıyoruz. Hangi grafik doğrudur, öğrenme oranını nasıl seçmeliyiz, doğru öğrenme oranını (alfayı) seçebilmiş miyiz? gibi soruların cevabını aşağıdaki grafiği inceleyerek bulmaya çalışabilirsiniz [3]. Düşük öğrenme oranına benzer bir grafik elde edildiyse, alfa büyütülmelidir veya çok yüksek öğrenme oranına benzer bir eğri elde edildiyse, alfa değeri azaltılmalıdır.

Not: Maliyet fonksiyonuna kayıp fonksiyonu (loss function) da denir.



Şekil 2.5 Öğrenme oranlarındaki değişimler

2.3 Adam Optimizasyon Algoritması

Adaptive Moment Estimation (Adam) algoritması [4], her iterasyonda öğrenme katsayısını güncelleyen yöntemlerdendir. Adadelta ve RMSProp yöntemlerinde olduğu gibi sadece önceki gradientlerin karelerini (vt) hesaba katmaz, ayrıca geçmiş gradientleri karesel olmadan da (mt) hesaba katar.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{4}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{5}$$

$$\acute{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{6}$$

$$\tilde{v} = \frac{v_t}{1 - \beta_2^t} \tag{7}$$

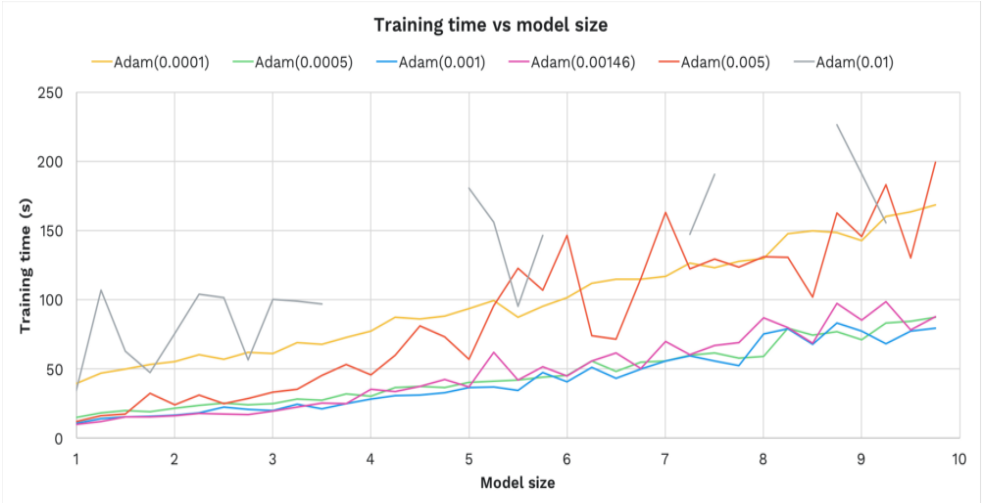
$$\theta_{t+1=\,\theta_t} - \frac{\alpha}{\sqrt{\tilde{v}_t + \epsilon}} . \acute{m}_t \tag{8}$$

Bu yöntem, Adagrad ve RMSProp yöntemlerinin avantajlarını bir araya getirecek şekilde tasarlanmıştır [5].

Adam, eğitim verilerine dayanarak yinelemeli ağ ağırlıklarını güncellemek için klasik stokastik gradyan iniş prosedürü yerine kullanılabilecek bir optimizasyon algoritmasıdır.

Adam algoritmasının avantajları şu şekilde sıralanabilir:

- Uygulaması basittir.
- Hesaplamalı olarak verimlidir.
- Hafıza ihtiyacı azdır.
- Gradientlerin köşegen ölçüsüne göre değişmez.
- Veri ve/veya parametreler açısından büyük olan problemler için çok uygundur.
- Sabit olmayan hedefler için uygundur.
- Çok gürültülü veya seyrek gradientlerle ilgili sorunlar için uygundur.
- Hiper parametrelerin sezgisel yorumu vardır ve genellikle çok az ayar gerektirir.



Şekil 2.6 Adam optimizasyon algoritmasının learning_rate değerlerinin karşılaştırılması

2.3.1 Adam Algoritmasının Çalışma Mantığı

Adam algoritması, klasik stokastik gradyan inişinden farklıdır. Stokastik gradyan inişi, tüm ağırlık güncellemeleri için tek bir öğrenme oranını (alfa olarak adlandırılır) tutar ve eğitim oranı, eğitim sırasında değişmez. Her ağ ağırlığı (parametre) için bir öğrenme hızı korunur ve ayrı ayrı öğrenme süreleri olarak uyarlanır. Bu yöntem, farklı parametreler için bireysel uyarlamalı öğrenme oranlarını, gradyanların birinci ve ikinci büyüklüklerinin tahminlerinden hesaplar. Yazarlar, Adam'ı diğer iki stokastik gradyan iniş uzantısının avantajlarını birleştirerek tanımlar.

Adaptive Gradient Algorithm (AdaGrad): Bu algoritma**,** seyrek gradientlerle ilgili sorunlarda (örneğin, doğal dil ve bilgisayarla görme sorunları) performansı artıran, parametre başına bir öğrenme oranı sağlar.

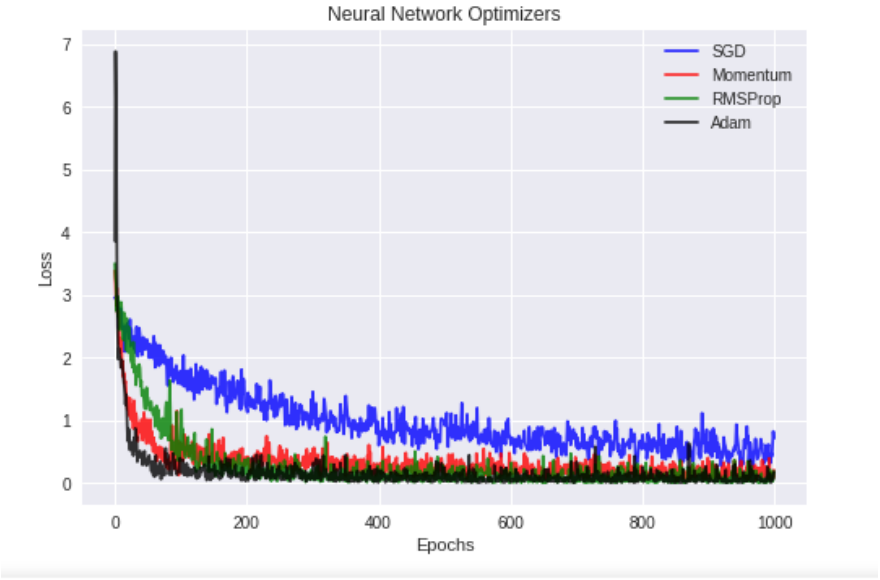
Root Mean Square Propagation (RMSProp): Bu algoritma, aynı zamanda ağırlık için gradyanların son büyüklüklerinin ortalamasına bağlı olarak uyarlanan parametre başına öğrenme oranlarını da korur (örneğin, ne kadar çabuk değişiyorsa). Bu durum, algoritmanın çevrimiçi ve durağan olmayan problemlerde (örneğin, gürültülü) iyi çalıştığı anlamına gelir. Adam, AdaGrad ve RMSProp'un faydalarını anlar ve birleştirir.

Parametre öğrenme oranlarını, RMSProp'taki gibi ortalama ilk büyüklüğe göre (ortalama) göre uyarlamak yerine, Adam algoritması ayrıca gradyanların ikinci büyüklüklerinin ortalamasını (merkezlenmemiş varyans) kullanır.

Spesifik olarak, algoritma gradyan ve kare gradyanın üstel bir hareketli ortalamasını hesaplar ve beta1 ile beta2 parametreleri, bu hareketli ortalamaların bozulma oranlarını kontrol eder.

Hareketli ortalamaların başlangıç değeri ve 1.0 (önerilen)'e yakın beta1 ve beta2 değerleri sıfır değer sapması sonucunu doğurur. Bu bias (önyargı), ilk önce önyargılı tahminleri hesapladıktan sonra önyargı düzeltilmiş tahminlerini hesaplayarak giderilir.

Adam algoritması derin öğrenme alanında popüler bir algoritmadır. Çünkü hızlı ve iyi sonuçlar ortaya çıkarır.



Şekil 2.7 Optimizasyon algoritmalarının kayıp değerlerinin karşılaştırılması

2.3.2 Adam Yapılandırma Parametreleri

Adam optimizasyon algoritmasının bazı parametreleri vardır. Bunları açıklayacak olursak;

Alpha: Öğrenme oranı veya adım büyüklüğü olarak da adlandırılır. Ağırlık oranları güncellenir (bu çalışmada, 0.001). Daha büyük değerlerde (örneğin, 0.3), ağırlık oranları güncellenmeden önce daha hızlı öğrenmeye neden olur. Bu da overfitting (aşırı öğrenme)'e neden olur. Daha küçük değerler (örneğin, 1.0E-5) egzersiz sırasında öğrenmeyi yavaşlatır ve underfitting (yavaş öğrenme)'e neden olur.

Beta1: İlk moment tahmini için üssel bozunma oranıdır (Bu çalışmada, 0.9).

Beta2: İkinci-büyüklük tahminleri için üssel azalış oranıdır (Bu çalışmada, 0.999). Bu değer, seyrek bir gradyan ile ilgili problemlerde (örneğin, NLP ve bilgisayarla görme problemleri) 1.0'a yakın olarak ayarlanmalıdır.

Epsilon: Uygulamada herhangi bir bölünmeyi sıfır ile önlemek için çok küçük bir sayıdır (Bu çalışmada, 1e-8).

Ayrıca öğrenme oranının azalması Adam algoritması ile de kullanılabilir.

```
def __init__(self, learning_rate=0.001, beta1=0.9, beta2=0.999, epsilon=1e-8,
             use_locking=False, name="Adam"):
```

Şekil 2.8 Projede kullanılan adam algoritmasının parametreleri

Popüler derin öğrenme kütüphanelerinin varsayılan parametreleri şu şekildedir [6]:

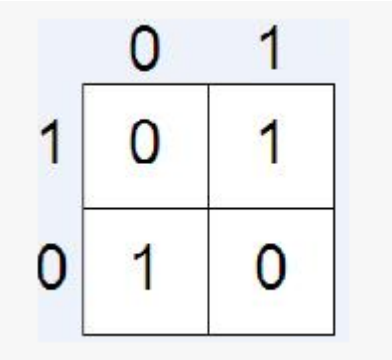
- **Tensorflow:** learning_rate=0.001, beta1=0.9, beta2=0.999, epsilon=1e-08.

  Keras: learning_rate=0.001, beta1=0.9, beta2=0.999, epsilon=1e-08, decay=0.0.
- **Blocks:** learning_rate=0.002, beta1=0.9, beta2=0.999, epsilon=1e-08, decay_factor=1.
- **Lasagne:** learning_rate=0.001, beta1=0.9, beta2=0.999, epsilon=1e-08.
- **Caffe:** learning_rate=0.001, beta1=0.9, beta2=0.999, epsilon=1e-08.
- **MxNet:** learning_rate=0.001, beta1=0.9, beta2=0.999, epsilon=1e-08.
- **Torch:** learning_rate=0.001, beta1=0.9, beta2=0.999, epsilon=1e-08.

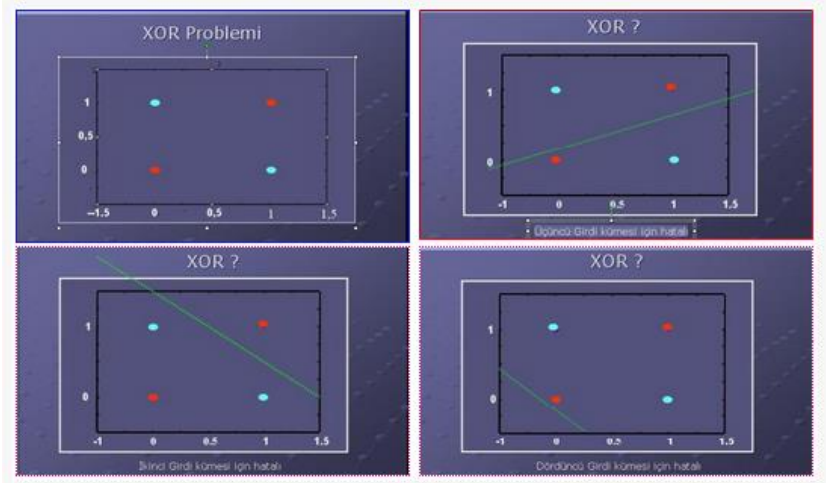
Bu projede tensorflow kütüphanesi kullanılmıştır ve varsayılan parametre değerleri yukarıda belirtilmiştir.

2.4 XOR Problemi

XOR fonksiyonu doğrusal ayrılabilir değildir. OR ve AND fonksiyonunun çıktıları düzlemde iki gruba ayrılabilir. Fakat XOR fonksiyonunda bu durum gerçekleştirilemez. Düzlemdeki çıktılar tek bir hatla ikiye bölünemez. En az iki doğru gerekiyor. XOR problemi Yapay Sinir Ağları'nın Hello World'ü olarak biliniyor. Perceptronlar XOR problemi gibi doğrusal olarak sınıflandırılamayan problemlerin çözümünde başarısızdır. XOR problemini çözmek için geriye yayılımlı çok katmanlı ağlardan faydalanılabilir.



Şekil 2.9 Basit XOR problemi



Şekil 2.10 Sınıflandırılamayan XOR problemi

2.5 Projede Kullanılan Ortam (IDE) ve Kütüphaneler

Çoğu zaman yapay zeka ile veri bilimi bir arada kullanılır fakat bazen birisinde çalışılıp, ötekisinde az etkinlik göstermesi olanaklıdır. Bu nedenle veri bilimi için önerilen dillerle yapay zeka için önerilen diller aynı olmayabilir. Bir başka konu da veri bilimi ya da yapay zeka için

kullanılmasa da bu alanlarda etkinlik gösterenlerin bilmesi gereken diller ve konular bulunmaktadır. Örneğin veri bilimi için öncelikle verinin elde edilmesi gerekir. Dolayısıyla veri erişimi ilgili dillerin bilinmesi gerekir. İşlenen verinin ne olduğunun da bilinmesi gerekir. İş dünyası ya da bilimsel verilerin kendisi üzerine de bilgili olunması gerekir. Bu projede de, en yaygın kullanılan veri bilimi dilleri arasında en çok sözü edilenlerden birisi olan Python programlama dili kullanılmıştır ve belli başlı kütüphaneler aracılığıyla veri bilimine katkı sağlanmıştır ve kolaylık elde edilmiştir.

2.5.1 Kullanılan Kütüphaneler

Veri bilimi için çeşitli kütüphaneler ön plana çıkmıştır ve bunların başında tensorflow gelmektedir. Bu proje için kullanılan bu kütüphaneyi ve diğer kütüphaneleri açıklayacak olursak;

2.5.1.1 Tensorflow

Aradan geçen kısa sürede tensorflow derin öğrenme konusunda kendisinden en çok söz ettiren ve yaygınlaşan derin öğrenme kütüphanesi olmuştur. C++ dili ile geliştirilen tensorflow'un bu dil dışında Python, Java gibi birçok programlama diline arayüz sağlaması, dağıtık olarak aynı anda birden fazla bilgisayarda çalışabilme olanakları ve son geliştirmelerle beraber Apple iOS telefon ve tabletlerde çalışabilmesinin sağlanması önemli artıları arasında görülüyor. Google'ın tanımıyla veri akış grafikleri kullanarak nümerik hesaplar yapabilen, açık kaynak kodlu bir kütüphanedir. Bu projede de yapay sinir ağları yöntemleri kullanılırken, hava durumu tahmini için tensorflow kütüphanesinden yararlanılmıştır. Yapılan bu projede Python diline arayüz sağlamıştır. Bu kütüphanenin birçok hazır metriklerinden yararlanılmıştır.

2.5.1.2 Matplotlib

Matplotlib kütüphanesi, grafik işlemleri ve iki boyutlu görselleştirme işlemleri için en çok kullanılan Python kütüphanesidir. İlk olarak John D. Hunter (JDH) tarafından geliştirilmiş olup, şimdi geniş bir yazılım ekibi tarafından geliştirilmekte ve yeni sürümleri ortaya çıkarılmaktadır. Özellikle Ipython gibi etkileşimli ve görsel projelere çok uygun bir kullanımı vardır. Bu projede de matplotlib kütüphanesi, grafik işlemleri ve iki boyutlu görselleştirmeler için kullanılmıştır.

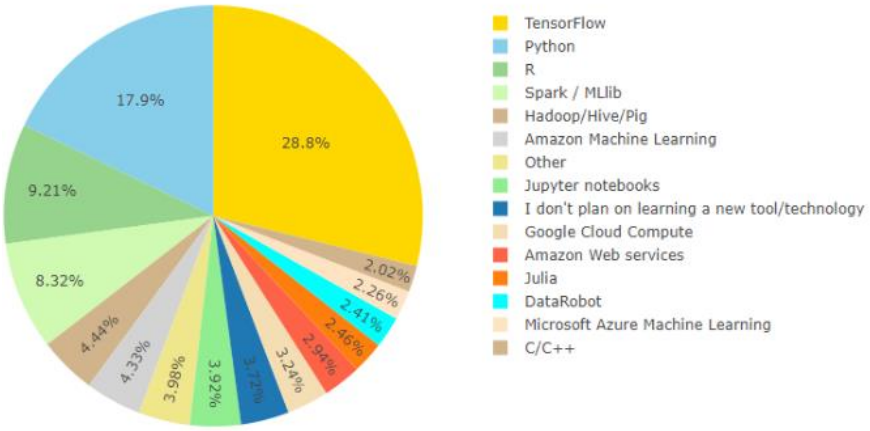
2.5.1.3 CSV

CSV formatında oluşturulan dosyaların sekmeli olarak veri okuma ve yazma işlemleri uygular. Excel tarafından oluşturulan bu dosyadan veri okuma ve yazma işlemlerinde hazır metotlar

sunar ve programcılar için okuma ve yazma işlemlerini kolaylaştırır. Bu projede de veri setinden verileri çekebilmek için csv kütüphanesi kullanılmıştır.

2.5.2 Kullanılan Ortam (IDE) ve Programlama Dili

Günümüzde veri bilimi için çeşitli programlama dilleri ön plana çıkmıştır ve bunların başında Python programlama dili gelmektedir. Tensorflow kütüphanesi de en çok tercih edilen kütüphanedir. Bu proje de Python programlama dili ile yazılmıştır ve Tensorflow kütüphanesi kullanılmıştır. Aynı zamanda bu proje, Visual Studio Code ortamında (idesinde) yazılmıştır.



Şekil 2.11 Günümüzde veri bilimi için en çok kullanılan platformlar

Python hakkında biraz bilgi verecek olursak;

Python 1990 yılında Guido van Rossum (2005-2012 yılları arasında Google'da çalışmıştır) tarafından Amsterdam'da geliştirilmeye başlanmış nesne yönelimli, yorumlanabilen, modüler ve etkileşimli bir programlama dilidir.

Python Yorumlanabilir: Python programları, çalışma zamanında yorumlayıcısı tarafından yorumlanır. Python kodunun derlenmesine ihtiyaç yoktur. Bu özelliği ile Perl ve Php dillerine benzerdir.

Python İnteraktif: Python etkileşimli konsola herhangi bir yerel bilgisayar veya Python yüklü sunucudan erişilebilir. Bir programlama ortamı kurduysanız, ortamı başlatabilir ve ilk önce o ortama girerek o ortamda kurmuş olduğunuz Python ve modüllerin sürümüne erişebilirsiniz.

Python Nesne Yönelimli: Python nesne yönelimli stil veya nesne içindeki kodu kapsülleyen programlama tekniğini destekler.

Python Yeni Başlayanlar İçin Bir Dil: Python, yeni başlayan programcılara yönelik harika bir dildir ve basit metin işleme, www tarayıcılar, oyunlar gibi geniş bir uygulama yelpazesinin geliştirilmesini destekler.

Python Dinamik, bytcode-derlenmiş Bir Dil: Kaynak kodunda değişkenlerin, parametrelerin, fonksiyonların veya yöntemlerin veri tipi bildirimleri yoktur. Bu, kodu kısa ve esnek hale getirir. Python, çalışma zamanında tüm değer türlerini izler ve çalıştırıldığında mantıklı olmayan kodu işaretler.

Kolay Öğrenilebilir: Python, insan diline benzer bir yapıya sahiptir, birkaç anahtar sözcük, basit yapı ve kolay tanımlanmış bir sözdizimi vardır. Python öğrenmek bu özelliklerinden dolayı çok kolaydır.

Kolay Okunabilir: Python kodları yalın söz diziminden dolayı kolayca okunabilir.

Geniş Bir Standart Kütüphane: Kütüphanenin Python'da toplu hali, UNIX, Windows ve Macintosh üzerinde çok taşınabilir ve çapraz platform uyumludur.

Taşınabilir: Python çok çeşitli donanım platformlarında çalışabilir ve tüm platformlarda aynı arabirime sahiptir.

Pek Çok IDE: Python ile kod yazarken kullanılabilecek çok fazla IDE (Integrated Development Enviroment) vardır: Eclipse, Pydev, Eric, Komodo IDE, PyCharm.

Çapraz Platform Desteği: Python çapraz (cross) platform desteği sayesinde birçok sistem üzerinde çalıştırılabilir. Pek çok Linux dağıtımının içerisinde Python 2.x sürümü yüklü gelmektedir. Ayrıca ülkemizde TUBİTAK tarafından geliştirilen Linux dağıtımı Pardus'un da bel kemiğini yine Python oluşturmaktadır. Popüler Linux dağıtımları da Python'u, çeşitli uygulamalarını geliştirmek için kullanmaktadırlar (Örnek olarak; Ubuntu Software Center).

Büyük Şirketler Kullanıyor: Python kullanarak masaüstü programlama, oyun programlama, taşınabilir cihaz programlama, web programlama ve ağ programlama çalışmaları rahatlıkla yürütülebilir.

Bellek Kullanımını Optimize Eder: Python, sahip olduğu Garbage Collector (çöp toplayıcı) sayesinde programın hafıza kullanımını optimize eder, programın kararlılığını ve performansını arttırır.

Diğer Diller İle Entegre: Python**,** Java ve .NET platformları ile entegre biçimde çalışma yeteneğine sahiptir.

2.6 Min-Max Normalizasyonu

Min-Max yöntemi, verileri doğrusal olarak normalize eder. Minimum; bir verinin alabileceği en düşük değer iken, maksimum; verinin alabileceği en yüksek değeri ifade eder. Bir veriyi Min-Max yöntemi ile 0 ve 1 aralığına indirgemek için aşağıdaki formül kullanılır.

$$X_{norm} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{9}$$

Bu eşitlikte;

X_{norm} = Normalize edilmiş veriyi,

X = Girdi değerini,

X_{min} = Girdi seti içerisinde yer alan en küçük sayıyı,

X_{max} = Girdi seti içerisinde yer alan en büyük sayıyı,

ifade etmektedir.

3 UYGULAMA ve DENEYSEL SONUÇLAR

Bu çalışmada, yapay sinir ağları yönteminin, tensorflow kütüphanesini de kullanarak veri bilimi üzerindeki etkisi görülmüştür. Hava durumu tahmini için kullanılan veri setinin bir kısmı Çizelge 3.1'de verilmiştir.

3.1 Veri Seti

Bu çalışmada kullanılan veri seti, İspanya'nın Madrid şehrindeki 1997 yılından, 2015 yılına kadar olan günlük hava durumu tahminlerini göstermektedir. Veri setinin tamamı gösterilemediği için bir kısmı aşağıda Çizelge 3.1'de verilmiştir.

Çizelge 3.1 YSA'da sıcaklık tahmini için kullanılacak olan veri setinden bir kısım

	CET	Max Tem	Mean Ter	Min Tem	Dew Poin	MeanDev	Min Dew	Max Hum	Mean Hu	Min Hum	Max Sea	Mean Se	Min Sea	Max Visil	Mean Vis	Min Visit	Max Win	Mean Wi	Max Gust	Precipitat	CloudCov	Events	WindDirDegrees
2	1997-1-1	7	4	2	5	3	2	100	95	76	1010	1008	1004	10	9	4	13	6		0	6		229
3	1997-1-2	7	3	0	6	3	0	100	92	71	1007	1003	997	10	9	4	26	8	47	0	5	Rain	143
4	1997-1-3	5	3	2	5	1	-1	100	85	70	1005	999	996	10	10	7	27	19		0	6	Rain-Sno	256
5	1997-1-4	7	3	-1	-2	-3	-4	86	63	49	1012	1010	1005	10	10	10	27	19	40	0	2		284
6	1997-1-5	2	0	-1	2	0	-3	100	95	86	1012	1008	1005	10	5	1	14	6		0	7	Snow	2
7	1997-1-6	7	3	1	2	-1	-3	100	82	57	1014	1010	1008	10	10	10	11	5		0	4		64
8	1997-1-7	2	0	-2	1	-1	-3	100	93	75	1016	1014	1009	10	7	0	6	2		0	7	Snow	43
9	1997-1-8	8	4	1	7	4	1	100	96	87	1015	1005	1003	10	8	4	26	8		0	7	Rain	273
10	1997-1-9	12	10	8	8	3	0	100	65	44	1015	1008	1003	10	10	10	48	23	48	0	4	Rain	274
11	1997-1-10	13	8	3	8	5	1	93	83	63	1021	1018	1016	10	10	10	13	5		0	5		100
12	1997-1-11	16	10	5	10	7	5	100	90	67	1024	1022	1021	10	6	0	11	2		0	3	Fog	344
13	1997-1-12	17	10	3	8	6	3	100	84	55	1024	1022	1020	10	10	10	14	8		0	1		345
14	1997-1-13	15	9	4	9	4	-1	100	75	38	1022	1021	1019	10	9	3	11	2		0	4	Fog	128
15	1997-1-14	11	6	0	4	2	0	100	81	50	1022	1020	1019	10	9	1	14	2		0	4		35
16	1997-1-15	11	8	5	7	5	4	100	83	62	1021	1019	1018	10	10	8	10	3		0	6	Rain	16
17	1997-1-16	12	9	6	8	7	6	100	89	72	1019	1017	1015	10	10	10	24	5		0	6		63
18	1997-1-17	13	9	5	9	5	1	100	78	47	1022	1018	1015	10	10	7	27	13		0	4	Rain	267
19	1997-1-18	6	3	2	6	4	2	100	98	93	1022	1019	1015	10	7	3	8	2		0	6	Rain	357
20	1997-1-19	9	7	6	7	6	5	100	97	87	1015	1011	1004	10	6	2	14	8		0	4	Rain	99
21	1997-1-20	9	7	5	9	7	5	100	100	100	1008	1005	1003	10	8	3	13	5		0	5	Rain	320
22	1997-1-21	10	8	6	7	6	5	100	89	71	1013	1011	1009	10	9	4	8	3		0	5		65
23	1997-1-22	13	8	3	8	5	3	100	85	58	1018	1016	1015	10	9	5	11	3		0	5	Rain	33
24	1997-1-23	12	10	9	10	9	9	100	96	82	1018	1014	1013	10	8	4	10	5		0	7	Rain	29
25	1997-1-24	13	10	7	9	8	7	100	90	67	1025	1021	1018	10	10	7	11	6		0	5	Rain	357
26	1997-1-25	12	9	7	9	7	6	100	89	76	1025	1024	1023	10	10	10	10	3		0	6		358
27	1997-1-26	14	9	5	8	6	5	100	83	55	1026	1025	1024	10	10	9	6	2		0	2		180
28	1997-1-27	15	8	2	8	6	2	100	84	55	1028	1027	1026	10	7	0	10	2		0	6	Fog	5
29	1997-1-28	14	10	6	7	4	2	93	74	48	1028	1027	1024	10	9	5	11	6		0	3		1
30	1997-1-29	11	8	5	7	6	4	100	83	67	1029	1027	1026	10	10	7	26	6		0	7		140
31	1997-1-30	15	11	8	8	7	5	100	80	59	1031	1030	1029	10	10	8	11	6		0	5		38

3.2 YSA Modelinin Yapısı ve Parametreleri

Yapay sinir ağı ile hava sıcaklığı tahmin işlemleri yapılırken sıcaklığı etkileyen etmenler meteorolojik olarak sıcaklık hesaplamada ortamda sıcaklığı etkileyen parametreler olarak aşağıda gösterildiği gibi 22 giriş parametresi vardır.

Giriş Parametreleri;

- Maksimum Sıcaklık (C)
- Ortalama Sıcaklık (C)
- Minimum Sıcaklık (C)

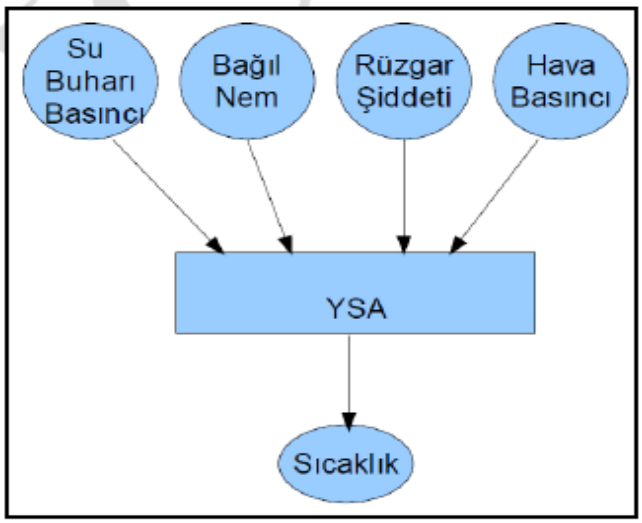
- Maksimum Çiy Derecesi (C)
- Ortalama Çiy Derecesi (C)
- Minimum Çiy Derecesi (C)
- Maksimum Nem
- Ortalama Nem
- Minimum Nem
- Maksimum Deniz Seviyesindeki Basınç (Pa)
- Ortalama Deniz Seviyesindeki Basınç (Pa)
- Minimum Deniz Seviyesindeki Basınç (Pa)
- Maksimum Görünürlük (Km)
- Ortalama Görünürlük (Km)
- Minimum Görünürlük (Km)
- Maksimum Rüzgar Hızı (Km/h)
- Ortalama Rüzgar Hızı (Km/h)
- Maksimum Fırtına Hızı (Km/h)
- Yağış (Mm)
- Soğuk Hava
- Yağış Durumu
- Rüzgar Dereceleri

Sistemin Çıkışı;

- Sıcaklık Dereceleri

Yukarıda verilen parametrelerin hava sıcaklığını etki ettikleri tespit edilmiştir. Bunların etkisi ile tasarlanacak YSA sisteminin vereceği çıkış olarak ise sıcaklık değerleri elde etmiş

olunacaktır. Bunun için tasarlanacak sistem için blok diyagramı aşağıda Şekil 3.1'de gösterilmiştir.

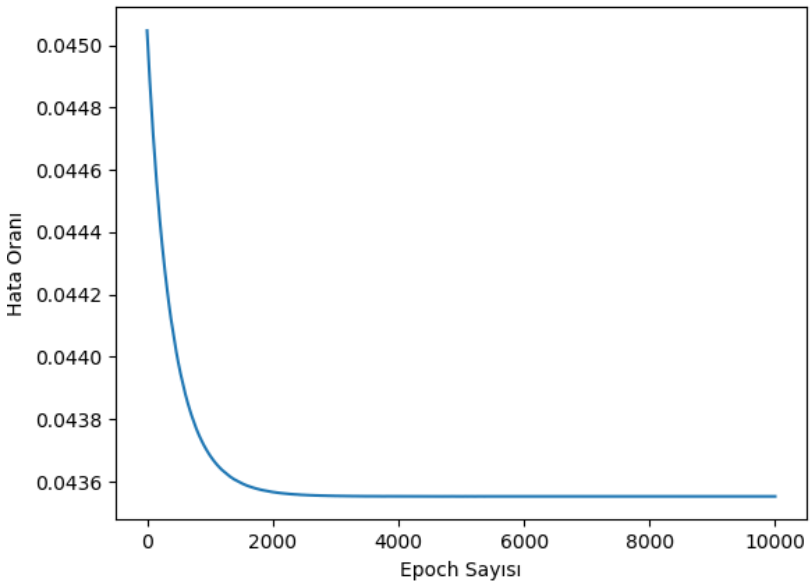


Şekil 3.1 Tasarlanan sistem için YSA blok diyagramından bir kısım

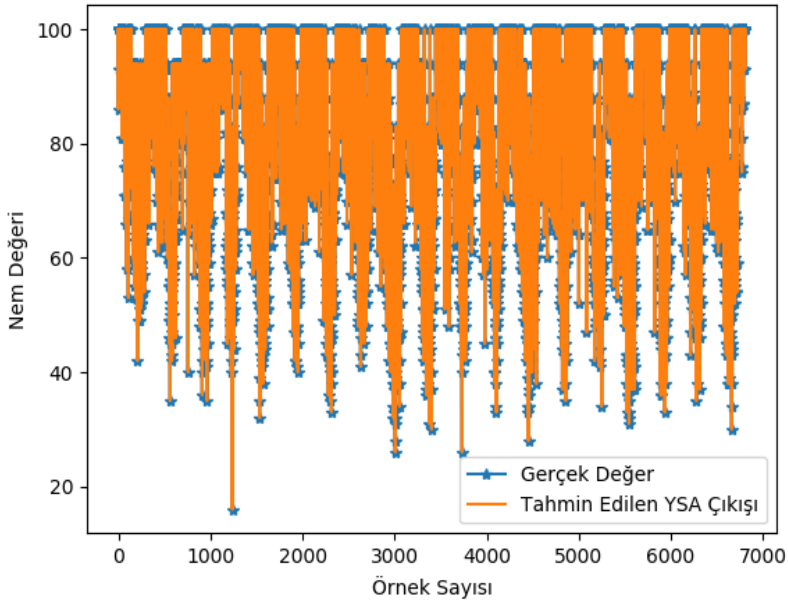
Blok diyagramında giriş parametrelerinin tamamı değil de bir kısmı verilmiştir. Bu diyagramdan da anlaşılacağı üzere sıcaklık tahmin işleminde belirli parametreler (su buhar basıncı, bağıl nem, rüzgar şiddeti, hava basıncı vs…) olmak üzere sistem 22 girişli tek çıkışlı şeklinde ileri beslemeli ve geri yayılımlı YSA modeli kullanılmıştır. Bu parametreler ilgili veriler, İspanya'nın Madrid şehrindeki meteoroloji istasyonundan günlük bazı eğitim parametreleri kullanılarak bu parametrelere karşılık tahmin işlemi gerçekleştirilmiştir.

3.3 Deneysel Sonuçlar

Bu projede, XOR problemini tensorflow kütüphanesi kullanarak yapay sinir ağları teknikleri ile çözmeye çalışan ve veri analizi olarak hava durumu tahmini yapılmaya çalışılmıştır ve elde edilen sonuçlar Şekil 3.2'de ve Şekil 3.3'te gösterilmiştir.



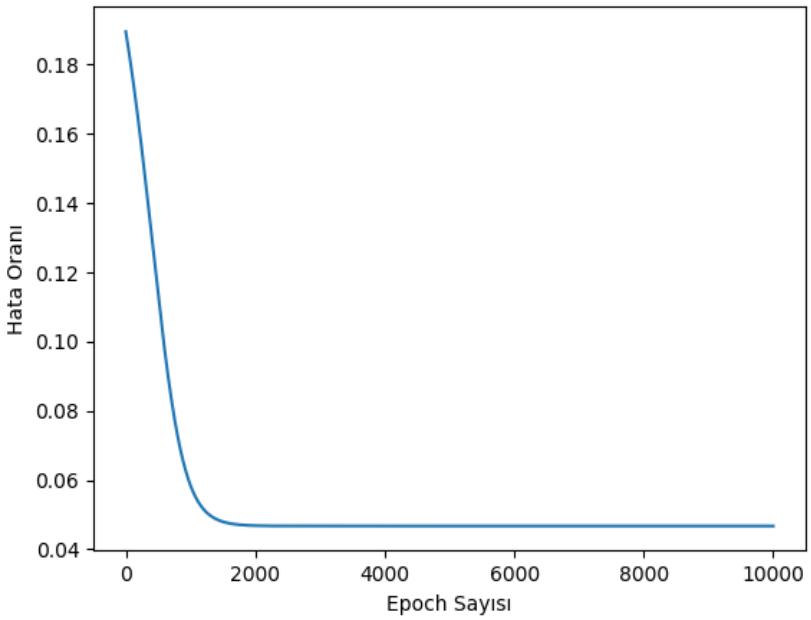
Şekil 3.2 Nem değerleri için 10000 epoch sonucundaki hata oranları



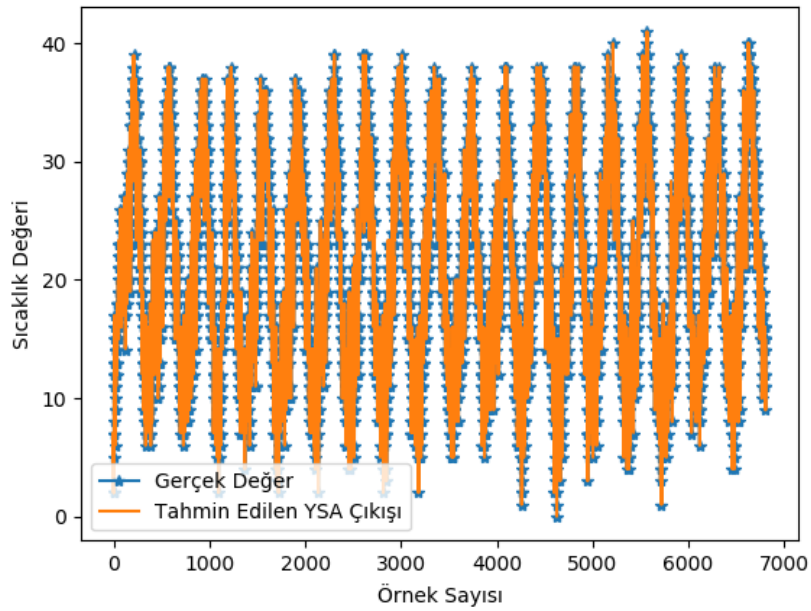
Şekil 3.3 Gerçek ve tahmin edilen nem değerleri

Yukarıdaki grafiklerde; 10000 epoch sayısı ile eğitilen yapay sinir ağının hata oranları ve bizim hava durumu veri setindeki örneklerin verilen giriş değerlerine göre çıkış değeri olan nem değerleri verilmiştir.

Yine aynı epoch sayısı ile örnek sayısı da sabit kalarak sıcaklık değerlerine bakacak olursak;



Şekil 3.4 Sıcaklık değerleri için 10000 epoch sonucundaki hata oranları



Şekil 3.5 Gerçek ve tahmin edilen sıcaklık değerleri

4 SONUÇ VE ÖNERİLER

Bu çalışmadaki temel amaç, yapay sinir ağlarını kullanarak en düşük hata ile XOR problemini de çözerek, Tensorflow kütüphanesi yardımıyla hava durumu tahmininde bulunmaktır. YSA kullanılarak elde edilen hava durumu tahmin işlemi çalışması için İspanya'nın Madrid şehrindeki 1997 yılından 2015 yılına kadar olan günlük hava durumu verilerinden (YSA giriş parametreleri ve çıkış parametreleri) yararlanılarak en az hata ile hava durumu tahmini gerçekleştirildi. Visual Studio Code, geliştirme ortamının işlevsel özellikler açısından güçlü ve kolayca genişletilebilir olması, geliştirilen yapay sinir ağı ile tahmin işlemi için kolaylık ve verimlilik sağlamıştır. Bununla birlikte bazı makine öğrenmesi ve yapay sinir ağları kütüphaneleri yazılım geliştirme ortamına dahil edilerek daha kısa sürede, daha iyi sonuçlar alınması sağlanmıştır. Nesne yönelimli olarak geliştirilmesi nedeniyle bazı sinir ağları modellerinin de eklenebilmesine olanak sağlanmıştır. Elde edilen karşılaştırmalar ve veriler sonucunda, uygulama aşamasında performans kriterleri açısından kullanımı engelleyecek durumların oluşmadığı söylenebilir. Yapılan bu çalışma ile Tensorflow kütüphanesinin yapay sinir ağlarına etkisi ve gelecekteki hava durumunun nasıl şekilleneceği görülmüş oldu. Burada ileri beslemeli ve geri yayılımlı YSA kullanıldı ve optimizasyon algoritması olarak Gradient Descent tabanlı algoritma çeşidi olan Adam optimizasyon algoritması kullanılarak bazı sonuçlar karşılaştırıldı. Python'da gerçekleştirilen bu yazılım günlük hayatta hava durumu tahmini ve daha farklı meteorolojik olayların tahmininde kullanılabilir.

KAYNAKLAR

[1] https://yapayzeka.ai/

[2] S. Ruder, "An overview of gradient descent optimization algorithms." *arXiv preprint arXiv:1609.04747*

[3] https://veribilimcisi.com/

[4] T. Tijmen, G. Hinton. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude." COURSERA: Neural networks for machine learning, 4.2 2012.

[5] D. P. Kingma, J. L. Ba, “Adam: a Method for Stochastic Optimization.” International Conference on Learning Representations,1–13, 2015.

[6] https://machinelearningmastery.com