**T.C.**

**SAKARYA ÜNİVERSİTESİ**

**BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ**

BSM 498 BİTİRME ÇALIŞMASI

YAPAY ZEKÂ İLE GOOGLE DİNOZOR OYUNU PROGRAMLAMA

B140910053-Yasemin ÇERÇİ

B141210026-Levent Can ŞENÇAMLAR

|  |  |  |
| --- | --- | --- |
| Fakülte Anabilim Dalı  Tez Danışmanı | :  : | BİLGİSAYAR MÜHENDİSLİĞİ  Dr.Öğr.Üyesi Seçkin ARI |

2018-2019 Bahar Dönemi

T.C.

SAKARYA ÜNİVERSİTESİ

**BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ**

YAPAY ZEKÂ İLE GOOGLE DİNOZOR OYUNU PROGRAMLAMA

BSM 498 - BİTİRME ÇALIŞMASI

Yasemin ÇERÇİ

Levent Can ŞENÇAMLAR

|  |  |  |
| --- | --- | --- |
| Fakülte Anabilim Dalı | : | BİLGİSAYAR MÜHENDİSLİĞİ |

Bu tez .. / .. / … tarihinde aşağıdaki jüri tarafından oybirliği / oyçokluğu ile kabul edilmiştir.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| ………………. | ………………. | ………………. |
| Jüri Başkanı | Üye | Üye |

**ÖNSÖZ**

Bu proje başlarken asıl amacımız yapay zeka algoritmalarını tanımaktı.Yapay zeka günümüzde hızla gelişen bir alan olması bizi bu alana yöneltti. Araştırmalara başladığımızda oyunlara yapay zeka entegre eden videolarla karşılaştık ve google dinozor oyununu programlamaya karar verdik.

**İÇİNDEKİLER**

|  |  |
| --- | --- |
| ÖNSÖZ……...................................................................................................... | iii |
| İÇİNDEKİLER.................................................................................................. | iv |
| SİMGELER VE KISALTMALAR LİSTESİ.................................................... | vi |
| ŞEKİLLER LİSTESİ......................................................................................... | vii |
| TABLOLAR LİSTESİ....................................................................................... | ix |
| ÖZET................................................................................................................. | x |
|  |  |
|  |  |
| BÖLÜM 1. |  |
| YAPAY SİNİR AĞLARI................................................................................... | 1 |
| * 1. YSA’ların Avantajları........................................................................ | 1 |
| * 1. YSA’ların Dezavantajları................................................................... | 3 |
| 1.3. Geleneksel Algoritmalar İle YSA’ların Karşılaştırılması.................. | 4 |
| 1.4. YSA’ların Kullanıldığı Alanlar......................................................... | 4 |
| 1.5. YSA’ların Sınıflandırılması.............................................................. | 5 |
| 1.5.1. Yapılarına göre YSA……………………………………………. | 6 |
| 1.5.2. Öğrenme algoritmalarına göre YSA........................................ | 6 |
| 1.5.3. Öğrenme zamanına göre YSA………………………………. | 7 |
| 1.6.YSA’nı Yapısı……………………………………………………… | 8 |
| 1.6.1. Biyolojik sinir hücresinin yapısı…………………………….. | 8 |
| 1.6.2. Yapay sinir hücresinin yapısı………………………………... | 8 |
| 1.6.3.YSA’nın yapısı………………………………………………. | 12 |
| 1.7. YSA’nı Eğitilmesi…………………………………………………. | 13 |
| 1.7.1. Katsayıların Belirlenmesi…………………………………. | 15 |
|  |  |
| BÖLÜM 2. |  |
| GENETİK ALGORİTMA…............................................................................. | 16 |
| 2.1.Temel Bilgiler GA’nın Uygulanması................................................. | 17 |
| 2.2. Genetik Algoritma Operatörleri........................................................ | 19 |
| 2.2.1.Çoğalma operatörü(Seçilme operatörü)................................. | 20 |
| 2.2.2. Çaprazlama operatörü(Crossover)......................................... | 22 |
| 2.2.3. Mutasyon operatörü(Mutation)…………………………… | 25 |
| 2.2.4. Durdurma kriteri………………………………………….. | 26 |
|  |  |
| BÖLÜM 3. |  |
| YSA VE GA’NIN GOOGLE DİNOZOR OYUNUNA ENTEGRE EDİLMESİ……………………………………………………………………. | 27 |
| 3.1. Yapay Sinir Ağı................................................................................. | 27 |
| 3.1.1. YSA girdileri ve çıktıları........................................................ | 28 |
| 3.2.Genetik Algoritma............................................................................. | 31 |
| 3.3. Yöntemin Detaylı İşleyişi………………………………………….. | 33 |
| 3.4. UML Diagramı…………………………….………………………. | 34 |
| 3.4.1. Dino model UML diagramı…………………………………. | 34 |
| 3.4.2. Dino game UML diagramı…………....…………………….. | 35 |
| 3.4.3. YSA UML diagramı………………………………..………. | 36 |
|  |  |
| BÖLÜM 4. |  |
| SONUÇLAR ………….…………………………………………..………...... | 37 |
| 4.1. Yaptığımız Denemeler…………………………………………….. | 37 |
|  |  |
|  |  |
| KAYNAKLAR……………………………………………………………….. | 41 |
| ÖZGEÇMİŞ……………………………………………….………………….. | 42 |
|  |  |
|  |  |
| BSM 498 BİTİRME ÇALIŞMASI DEĞERLENDİRME VE SÖZLÜ SINAV TUTANAĞI………………………………………………………………… | 43 |
|  |  |

SİMGELER VE KISALTMALAR LİSTESİ

|  |  |
| --- | --- |
| ABD | : Amerika Birleşik Devletleri |
| AR-GE | : Araştırma Geliştirme |
| BM | : Birleşmiş Milletler |
| GA | : Genetik Algoritma |
| Ör | : Örnek |
| vb. | : ve benzeri |
| YSA | : Yapay Sinir Ağları |
| YZ | : Yapay Zekâ |

**ŞEKİLLER LİSTESİ**

|  |  |  |
| --- | --- | --- |
| Şekil 1.1. | YSA yapısı……………………………................................ | 6 |
| Şekil 1.2. | YSA sinir hücresi………………………………………….. | 9 |
| Şekil 1.3. | YSA yapısı2………………………………………...……... | 12 |
| Şekil 2.1. | Genetik algoritmanın akış diyağram….…………...………. | 18 |
| Şekil 2.2. | Rulet tekerleği……………………………………………... | 20 |
| Şekil 2.3. | Truva yöntemi……………………………………………... | 21 |
| Şekil 2.4. | Sabit durum yöntemi………………………………………. | 21 |
| Şekil 2.5. | Sıralama yöntemi…………………………………………... | 22 |
| Şekil 2.6. | CX çaprazlama…………………………………………….. | 25 |
| Şekil 3.1. | YSA yapıcı method……………………………...………… | 27 |
| Şekil 3.2. | Weight ve bias…………………………………………….. | 27 |
| Şekil 3.3. | Aktivasyon fonksiyonu……………………………………. | 28 |
| Şekil 3.4. | Temsili YSA modeli…………………………………..…… | 29 |
| Şekil 3.5. | YSA’da kullanılacak katmanlardaki nöron sayısının belirlenmesi……………………………………………….. | 30 |
| Şekil 3.6. | Dinozorun eğilme ve zıplaması……………………………. | 30 |
| Şekil 3.7. | Engel bilgileri……………………………………………… | 30 |
| Şekil 3.8. | Dinozorları skoruna göre sıralma……………………..…… | 31 |
| Şekil 3.9. | Yüksek skorun değişip değişmeyeceğini belirleyen kod….. | 31 |
| Şekil 3.10. | En iyi klonun atanması…………………………………….. | 31 |
| Şekil 3.11. | Çaprazlama metodu………………………………………... | 31 |
| Şeki 3.12. | Bias mutasyonlama metodu……………………………….. | 32 |
| Şekil 3.13. | Weight mutasyonlama metodu…………………………….. | 32 |
| Şekil 3.14. | Weight ve biaslara random değer atama…………………... | 32 |
| Şekil 3.15. | Yöntemin detaylı işleyişi…………………………………... | 33 |
| Şekil3.16. | Dino model Uml…………………………………................ | 34 |
| Şekil3.17. | Dino game Uml…………………………………………….. | 35 |
| Şekil3.18. | YSA Uml…………………………………………………... | 36 |
| Şekil 4.1. | 50 ve 100 lük 2 gizli katman 40 tekrar GA yok | 37 |
| Şekil 4.2. | 30luk tek gizli katman GA var | 38 |
| Şekil 4.3. | 30luk tek gizli katman 100 jenerasyon GA var | 38 |
| Şekil 4.4. | 50 ve 100lük 2 gizli katman 40 jenerasyon | 39 |
| Şekil 4.5. | 75 ve 100lük 2 gizli katman 40 jenerasyon oyun hızı 13 | 39 |
| Şekil 4.6. | 100 ve 150lik 2 gizli katman 40 jenerasyon oyun hızı 15 | 40 |
|  |  |  |

**TABLOLAR LİSTESİ**

|  |  |  |
| --- | --- | --- |
| Tablo 1.1. | YSA ve geleneksel algoritmaların karşılaştırılması……….. | 4 |
| Tablo 1.2. | Bazı toplama fonksiyonları ……..………………………… | 10 |
| Tablo 1.3. | Aktivasyon fonksiyonları………………………………..… | 11 |
|  |  |  |

**ÖZET**

**Anahtar kelimeler:** Yapay Sinir Ağları, Genetik Algoritma, Deep Learning, Machine Learning, Yapay Zeka

Bu projeye başlarken asıl amacımız yapay zeka algoritmalarını tanımaktı. Araştırmalara başladığımızda oyunlara yapay zeka entegre eden videolarla karşılaştık ve google dinozor oyununu programlamaya karar verdik. Projemizde genel bir ifade ile yapay sinir ağları ve genetik algoritma kullanılmıştır.

Derin öğrenme bir veya daha fazla katman içeren yapay sinir ağları ve benzeri makine öğrenme algoritmalarını kapsayan çalışma alanıdır. Bizde projemizde yapay sinir ağlarını kullanarak dinozor popülasyonunun önüne çıkacak olan engelleri aşıp, en iyi puanı almasını hedefledik. Yapay sinir ağımız 3 katmandan meydana gelmektedir. İlk katman engellerin mesafesi yüksekliği gibi değerleri alıp gizli katmana iletilir, çıkış katmanında ise dinozorların atlama veya eğilmesini belirlemektedir. Dinozorlar öldüğünde genetik algoritma kullanılarak son iki dinozorun genleri bir sonraki nesilde dinozorlara aktarılarak iyileştirme hedeflenmiştir. Bu şekilde dinozorlar daha uzun süre hayatta kalarak daha iyi puanlar elde edilir.

# YAPAY SİNİR AĞLARI

Yapay sinir ağları (YSA), insan beyninin özelliklerinden olan öğrenme yolu ile yeni bilgiler türetebilme, yeni bilgiler oluşturabilme ve keşfedebilme gibi yetenekleri, herhangi bir yardım almadan otomatik olarak gerçekleştirebilmek amacı ile geliştirilen bilgisayar sistemleridir. Yapay sinir ağları; insan beyninden esinlenerek, öğrenme sürecinin matematiksel olarak modellenmesi uğraşı sonucu ortaya çıkmıştır. Bu nedenledir ki, bu konu üzerindeki çalışmalar ilk olarak beyni oluşturan biyolojik üniteler olan nöronların modellenmesi ve bilgisayar sistemlerinde uygulanması ile başlamış, daha sonraları bilgisayar sistemlerinin gelişimine de paralel olarak bir çok alanda kullanılır hale gelmiştir. İnsan beyninin çalışma prensibini taklit ederek çalışan bu sistemler, her ne kadar bilgisayar teknolojisi hızlı bir gelişim göstermiş, işlem hızları nano saniyeler mertebesine inmiş olsa da, bırakalım insan beynini, ilkel bir canlı beyninin fonksiyonları dahi baz alındığında, böyle bir organizmanın yanında çok ilkel kalmaktadır. Nano saniyeler bazındaki işlem hızları ile YSA'lar, mili saniyeler mertebesindeki işlen hızları ile işlem yapan insan beyninin işlevselliğinin henüz çok uzağındadır. Burada kısa bir hatırlatma yapmak gerekirse; insan beyninde yaklaşık 10¹¹ sinir hücresinin varlığından bahsedilmekle birlikte, bu sayının bilgisayar ortamında modellenmesi şu an için mümkün görünmemektedir. Fakat karar hızı açısından insan beyni ile henüz yarışamasalar bile, YSA'lar yapısallıkları ve hassas eşleştirmelerin başarı ile gerçekleştirebilmeleri ile gün geçtikçe daha fazla uygulama alanı bulmaktadır.

**1.1YSA’ların Avantajları**

YSA'lar, uygulanan ağ modeline göre değişik karakteristik özellikler göstermelerine karşın temel birkaç ortak özelliğe sahiptirler.

1. YSA lar bir çok hücreden meydana gelir ve bu hücreler eş zamanlı olarak çalışarak karmaşık işlevleri yerine getirir. Diğer bir değişle karmaşık işlevler bir çok hücrenin eş zamanlı çalışması ile meydana getirilir. Süreç içerisinde bu hücrelerden her hangi biri işlevini yitirse dahi sistem güvenli bir şekilde çalışmasına devam edebilir.

2. Eğitim esnasında kullanılan nümerik bilgilerden, problemin genel özellikleri elde etmesi ve böylelikle eğitim sırasında kullanılmayan girdiler için de, anlamlı yanıtlar üretebilmesidir.

3. Yapı üzerinde dağılmış belli tipteki non-lineer alt birimler, non lineer problemlerin de çözümünü mümkün kılmaktadır.

4. YSA'lar makina öğrenmesi gerçekleştirebilirler. Yapay sinir ağlarının temel işlevi zaten bilgisayarın öğrenmesini sağlamaktır. Olayları öğrenerek benzer olaylar karşısında mantıklı kararlar verebilirler.

5. Bilgi işleme yöntemleri geleneksel programlamadan farklıdır. Bu nedenle geleneksel programlamanın getirdiği bir çok olumsuzluk ortadan kaldırılabilir

6. Bilgiler ağın tamamında saklanır. Geleneksel programlamada olduğu gibi bilgiler veri tabanları yada dosyalarda belli bir düzende tutulmaz, ağın tamamına yayılarak değerler ile ölçülen ağ bağlantılarında saklanmaktadır. Hücrelerden bazılarının işlevini yitirmesi, anlamlı bilginin kaybolmasına neden olmaz.

7. Dağıtık belleğe sahiptirler. YSA'larda bilgi ağa dağılmış bir şekilde tutulur. Hücrelerin bağlantı ve ağırlık dereceleri, ağın bilgisini gösterir. Bu nedenle tek bir bağlantının kendi başına anlamı yoktur.

8. Örnekleri kullanarak öğrenirler. YSA'nın öğrenebilmesi için örneklerin belirlenmesi, bu örneklerin ağa gösterilerek istenen çıktılara göre ağın eğitilmesi gerekmektedir. Ağın başarısı, seçilen örnekler ile doğru orantılıdır, ağa olay bütün yönleri ile gösterilemezse ağ yanlış çıktılar üretebilir.

9. Daha önce görülmemiş örnekler hakkında bilgi üretebilirler. YSA'lar eğitimleri sırasında kendilerine verilen örneklerden genellemeler çıkarırlar ve bu genellemeler ile yeni örnekler hakkında bilgi üretebilirler.

10. Algılamaya yönelik olaylarda kullanılabilirler. YSA'ların en başarılı oldukları alanlar, algılamaya yönelik uygulama alanlarıdır. Bu alanlarda başarıları kanıtlanmıştır.

11. Örüntü (pattern) ilişkilendirme ve sınıflandırma yapabilirler. YSA'lar kendilerine örnekler halinde verilen örüntüleri kendisi veya diğerleri ile ilişkilendirebilir. Ayrıca kendisine verilen örneklerin kümelenmesi ile, bir sonraki verinin hangi kümeye dahil olacağının karar verilmesi konusunda kullanılabilirler.

12. Örüntü tamamlama yapabilirler. Ağa eksik bilgileri içeren örüntüler verildiğinde eksik bilgilerin tamamlanması konusunda başarılıdırlar.

13. Kendi kendine öğrenebilme ve organize etme yetenekleri vardır. YSA'lar online olarak öğrenebilirler ve kendi kendilerini eğitebilirler. takimata sanctus est Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut lab ore sit et dolore magna.

14. Eksik bilgi ile çalışabilmektedirler. Geleneksel sistemlerin aksine YSA'lar eğitildikten sonra veriler eksik bilgi içerse dahi, çıktı üretebilirler. Bu durum bir performans kaybı yaratmaz, performans kaybı eksik bilginin önemine bağlıdır. Burada bilgilerin önem dereceleri eğitim sırasında öğrenilir.

15. Hata töleransına sahiptirler. YSA'ların eksik bilgilerle çalışabilmeleri ve bazı hücreleri bozulsa dahi çalışabilmeleri, onları hatalara karşı töleranslı yapar.

16. Dereceli bozulma (Graceful degradation) gösterirler. Bir ağ, zaman içerisinde yavaş ve göreceli bir bozulmaya uğrar. Ağlar problemin ortaya çıktığı anda hemen bozulmazlar.

**1.2. YSA’ların Dezavantajları**

YSA'ların, pek çok avantajın yanında bazı dezavantajları da vardır. Belli başlı dezavantajları;

1. Donanım bağımlıdır. YSA'ların en önemli sorunu donanım bağımlı olmalarıdır. YSA'ların en önemli özellikleri ve var oluş nedenlerinden birisi olan paralel işlem yapabilme yeteneği, paralel çalışan işlemciler ile performans gösterir.

2. Uygun ağ yapısının belirlenmesinde belli bir kural yoktur. YSA'larda probleme uygun ağ yapısının belirlenmesi için geliştirilmiş bir kural yoktur. Uygun ağ yapısı deneyim ve deneme yanılma yolu ile belirlenmektedir.

3. Ağın parametre değerlerinin belirlenmesinde de belli bir kural yoktur. YSA'larda öğrenme katsayısı, hücre sayısı, katman sayısı gibi parametrelerin belirlenmesinde belirli bir kural yoktur. Bu değerlerin belirlenmesi için belirli bir standart olmamakla birlikte her problem için farklı bir yaklaşım söz konusu olabilmektedir.

4. Öğrenilecek problemin ağa gösterimi önemli bir problemdir. YSA'lar nümerik bilgiler ile çalışabilmektedirler. Problemler YSA'lara tanıtılmadan önce nümerik değerlere çevrilmek zorundadırlar. Burada belirlenecek gösterim mekanizması ağın performansını doğrudan etkileyecektir. Bu da kullanıcının yeteneğine bağlıdır.

5. Ağın eğitiminin ne zaman bitirilmesi gerektiğine ilişkin belli bir yöntem yoktur. Ağın örnekler üzerindeki hatasının belirli bir değerin altına indirilmesi eğitimin tamamlandığı anlamına gelmektedir. Burada optimum neticeler veren bir mekanizma henüz yoktur ve YSA ile ilgili araştırmaların önemli bir kolunu oluşturmaktadır.

6. Ağın davranışları açıklanamamaktadır. Bu sorun YSA'ların en önemli sorunudur. YSA bir probleme çözüm ürettiğizaman, bunun neden ve nasıl olduğuna ilişkin bir ipucu vermez. Bu durum ağa olan güveni azaltıcı bir unsurdur.

**1.3**. **Geleneksel Algoritmalar ile YSA'ların Karşılaştırılması**

Tablo 1.1. YSA ve geleneksel algoritmaların karşılaştırması

|  |  |
| --- | --- |
| Geleneksel Algoritmalar | Yapay Sinir Ağları |
| Çıkışlar, koyulan kurallara girişlerin uygulanması ile elde edilir. | Öğrenme esnasında giriş çıkış bilgileri verilerek, kurallar koyulur. |
| Hesaplama; merkezi, eş zamanlı ve ardışıldır. | Hesaplama; toplu, eş zamansız ve öğrenmeden sonra paraleldir. |
| Bellek paketlenmiş ve hazır bilgi depolanmıştır. | Bellek ayrılmış ve ağa yayılmıştır. |
| Hata töleransı yoktur. | Hata töleransı vardır. |
| Nispeten hızlıdır | Yavaş ve donanıma bağımlıdır. |
| Bilgiler ve algoritmalar kesindir. | Deneyimden yararlanır. |

**1.4. YSA’ların Kullanıldığı Alanlar**

Yapay sinir ağları başlıca; Sınıflandırma, Modelleme ve Tahmin uygulamaları olmak üzere, pek çok alanda kullanılmaktadır. Başarılı uygulamalar incelendiğinde, YSA'ların çok boyutlu, gürültülü, karmaşık, kesin olmayan, eksik, kusurlu, hata olasılığı yüksek sensör verilerinin olması ve problemi çözmek için matematiksel modelin ve algoritmaların bulunmadığı, sadece örneklerin var olduğu durumlarda yaygın olarak kullanıldıkları görülmektedir. Bu amaçla geliştirilmiş ağlar genellikle şu fonksiyonları gerçekleştirmektedirler; Muhtemel fonksiyon kestirimleri, Sınıflandırma, İlişkilendirme veya örüntü eşleştirme, Zaman serileri analizleri, Sinyal filtreleme, Veri sıkıştırma, Örüntü tanıma, Doğrusal olmayan sinyal işleme, Doğrusal olmayan sistem modelleme, Optimizasyon, Kontrol YSA'lar pek çok sektörde değişik uygulama alanları bulmuştur. Bunlardan bazıları;

**Uzay:** Uçuş simülasyonları, otomatik pilot uygulamaları, komponentlerin hata denetimleri vs.

**Otomotiv:** Otomatik yol izleme, rehber, garanti aktivite analizi, yol koşullarına göre sürüş analizi vs.

**Bankacılık:** Kredi uygulamaları geliştirilmesi, müşteri analizi ve kredi müraacat değerlendirilmesi, bütçe yatırım tahminleri vs.

**Savunma:** Silah yönlendirme, hedef seçme, radar, sensör sonar sistemleri, sinyal işleme, görüntü işleme vs.

**Elektronik:** Kod sırası öngörüsü, çip bozulma analizi, non-lineer modelleme vs. **Eğlence:** Animasyonlar, özel efektler, pazarlama öngörüsü vs.

**Finans:** Kıymet biçme, pazar performans analizi, bütçe kestirimi, hedef belirleme vs. Sigortacılık: ürün optimizasyonu, uygulama politikası geliştirme vs.

**Üretim:** üretim işlem kontrolü, ürün dizaynı, makina yıpranmalarının tespiti, dayanıklılık analizi, kalite kontrolü, iş çizelgeleri hazırlanması vs.

**Sağlık:** göğüs kanseri erken teşhis ve tedavisi, EEG, ECG, MR, kalite artırımı, ilaç etkileri analizi, kan analizi sınıflandırma, kalp krizi erken teşhis ve tedavisi vs. Petro kimya: arama, verim analizi vs.

**Robotik:** yörünge kontrol, forklift robotları, görsel sistemler, uzaktan kumandalı sistemler, optimum rota belirleme vs.

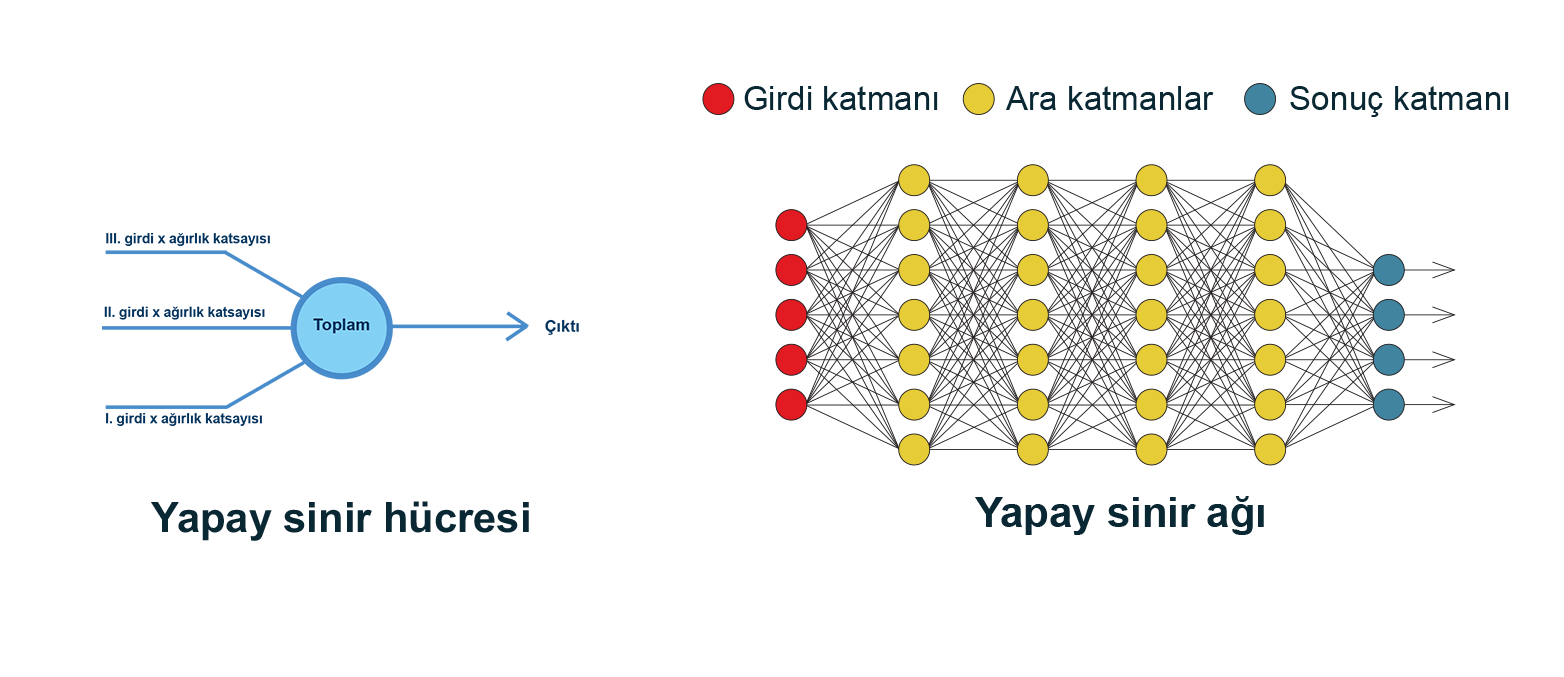
**Dil:** sözcük tanıma, yazı ve konuşma çevrimi, dil tercüme vs. Telekomünikasyon: görüntü ve data karşılaştırma, filtreleme, eko ve gürültü sönümlendirilmesi, ses ve görüntü işleme, trafik yoğunluğunun kontrolü ve anahtarlama vs.

**Güvenlik:** parmak izi tanıma, kredi kartı hileleri saptama, retina tarama, yüz eşleştirme vs.

Bu örnekler çoğaltılabilir. Görüldüğü gibi YSA'lar günlük hayatımızda farkında olmadığımız pek çok alanda kullanılmaktadır. Gün geçtikçe uygulama alanları genişlemekte ve gelişmektedir.

**1.5. YSA’ların Sınıflandırılması**

Yapay sinir ağları işleyiş olarak benzer olmalarına rağmen herhangi bir tasarım ve işleyiş standardı bulunmamaktadır. Nöron dizilimlerine, nöronların ağırlıklarının düzenleme için yapılan hesaplamaların türüne ve zamanına göre yapay sinir ağlarını üç ayrı dalda inceleyebiliriz.



Şekil 1.1. YSA yapısı

**1.5.1 Yapılarına göre YSA**

Yapay sinir ağları içerdiği nöronların birbirine bağlanış şekline göre ileri ve geri beslemeli olarak ikiye ayrılır.

İleri Beslemeli Ağlar: İleri beslemeli ağlarda nöronlar girişten çıkışa doğru düzenli katmanlar şeklindedir. Bir katmandan sadece kendinden sonraki katmanlara bağ bulunmaktadır. Yapay sinir ağına gelen bilgiler giriş katmanına daha sonra sırasıyla ara katmanlardan ve çıkış katmanından işlenerek geçer ve daha sonra dış dünyaya çıkar.

Geri Beslemeli Yapay Sinir Ağları: Geri beslemeli yapay sinir ağlarında ileri beslemeli olanların aksine bir hücrenin çıktısı sadece kendinden sonra gelen hücrenin katmanına girdi olarak verilmez. Kendinden önceki katmanda veya kendi katmanında bulunan herhangi bir hücreye de girdi olarak bağlanabilir. Bu yapısı ile geri beslemeli yapay sinir ağları doğrusal olmayan dinamik bir davranış göstermektedir. Geri besleme özelliğini kazandıran bağlantıların bağlanış şekline göre geri aynı yapay sinir ağıyla farklı davranışta ve yapıda geri beslemeli yapay sinir ağları elde edilebilir.

**1.5.2 Öğrenme algoritmalarına göre YSA**

Yapay sinir ağlarının verilen girdilere göre çıktı üretebilmesinin yolu ağın öğrenebilmesidir. Bu öğrenme işleminin de birden fazla yöntemi vardır. Yapay sinir ağları öğrenme algoritmalarına göre danışmanlı, danışmansız ve takviyeli öğrenme olarak üçe ayrılır.

Danışmanlı Öğrenme: Danışmanlı öğrenme sırasında ağa verilen giriş değerleri için çıktı değerleri de verilir. Ağ verilen girdiler için istenen çıkışları oluşturabilmek için kendi ağırlıklarını günceller. Ağın çıktıları ile beklenen çıktılar arasındaki hata hesaplanarak ağın yeni ağırlıkları bu hata payına göre düzenlenir. Hata payı hesaplanırken ağın bütün çıktıları ile beklenen çıktıları arasındaki fark hesaplanır ve bu farka göre her hücreye düşen hata payı bulunur. Daha sonra her hücrenin kendine gelen ağırlıkları günceller.

Danışmansız Öğrenme: Danışmasız öğrenmede ağa öğrenme sırasında sadece örnek girdiler verilmektedir. Herhangi bir beklenen çıktı bilgisi verilmez. Girişte verilen bilgilere göre ağ her bir örneği kendi arasında sınıflandıracak şekilde kendi kurallarını oluşturur. Ağ bağlantı ağırlıklarını aynı özellikte olan dokuları ayırabilecek şekilde düzenleyerek öğrenme işlemini tamamlar.

Destekleyici Öğrenme: Bu öğrenme yaklaşımında ağın her iterasyonu sonucunda elde ettiği sonucun iyi veya kötü olup olmadığına dair bir bilgi verilir. Ağ bu bilgilere göre kendini yeniden düzenler. Bu sayede ağ herhangi bir girdi dizisiyle hem öğrenerek hem de sonuç çıkararak işlemeye devam eder. Örneğin satranç oynayan bir yapay sinir ağı yaptığı hamlenin iyi veya kötü olduğunu anlık olarak ayırt edememesine rağmen yine de hamleyi yapar. Eğer oyun sonuna geldiğinde program oyunu kazandıysa yaptığı hamlelerin iyi olduğunu varsayacaktır ve bundan sonraki oyunlarında benzer hamleleri iyi olarak değerlendirerek oynayacaktır.

**1.5.3. Öğrenme zamanına göre YSA**

Yapay sinir ağları öğrenme zamanına göre de statik ve dinamik öğrenme olarak ikiye ayrılır.

Statik Öğrenme: Statik öğrenme kuralıyla çalışan yapay sinir ağları kullanmadan önce eğitilmektedir. Eğitim tamamlandıktan sonra ağı istenilen şekilde kullanılabilinir. Ancak bu kullanım sırasında ağın üzerindeki ağırlıklarda herhangi bir değişiklik olmaz.

Dinamik Öğrenme: Dinamik öğrenme kuralı ise yapay sinir ağlarının çalıştığı süre boyunca öğrenmesini öngörerek tasarlanmıştır. Yapay sinir eğitim aşaması bittikten sonra da daha sonraki kullanımlarında çıkışların onaylanmasına göre ağırlıklarını değiştirerek çalışmaya devam eder.

**1.6. YSA'nın Yapısı**

**1.6.1 Biyolojik sinir hücresinin yapısı**

Bir insanın beyninde yaklaşık olarak 10 milyar sinir hücresi ve bu nöronların birbirleriyle yaptığı bağlantı sayısının ise 60 trilyon olduğu tahmin edilmektedir. Bu sinirler girdi bilgilerini duyu organlarından alırlar. Daha sonra alıcı (taşıyıcı) sinirler bu sinyalleri işleyip bir sonraki sinire aktararak sinyalin merkezi sinir sistemine kadar ulaşmasını sağlar. Merkezi sinir sistemi bu sinyalleri alıp yorumladıktan sonra tepki sinyallerini üretir. Bu sinyaller de tepkilerin oluşacağı organlara tepki sinirleri vasıtasıyla iletilir. Bu sayede duyu organlarından gelen bilgilere karşı tepki organlarına uygun işaretler sinir sistemi vasıtasıyla yollanır.

Yapay sinir ağları biyolojik sinir ağlarının modellemesi olduğu için, öncelikle biyolojik sinir sisteminin yapısına bakmak gerekir. Biyolojik sinir sisteminin temel yapı taşı olan nöronların yapısı dört ana bölümden oluşmaktadır; dendrit, akson, çekirdek ve bağlantılar. Dendritlerin sinir hücresinin ucunda bulunan ve ağaç kökü görünümüne sahip bir yapıya sahiptir. Dendritlerin görevi bağlı olduğu diğer nöronlardan veya duyu organlarından gelen sinyalleri çekirdeğe iletmektir. Çekirdek dendrit tarafından gelen sinyalleri bir araya toplayarak ve aksona iletir. Toplanan bu sinyaller akson tarafından işlenerek nöronun diğer ucunda bulunan bağlantılara gönderilir. Bağlantılar ise yeni üretilen sinyalleri diğer nöronlara iletir.

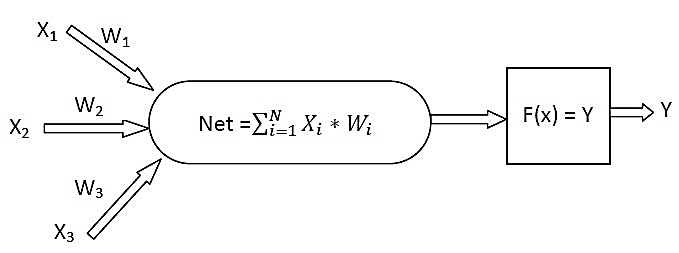
Şekil 1. Sinir hücresi ve Sinir sisteminin çalışma prensibi.

**1.6.2 Yapay sinir hücresinin yapısı**

Yapay sinir hücreleri de biyolojik sinir hücrelerine benzer yapıdadır. Yapay nöronlar da aralarında bağ kurarak yapay sinir ağlarını oluştururlar. Aynı biyolojik nöronlarda olduğu gibi yapay nöronların da giriş sinyallerini aldıkları, bu sinyalleri toplayıp işledikleri ve çıktıları ilettikleri bölümleri bulunmaktadır.

Bir yapay sinir hücresi beş bölümden oluşmaktadır;

* Girdiler
* Ağırlıklar
* Toplama Fonksiyonu (Birleştirme Fonksiyonu)
* Aktivasyon fonksiyonu
* Çıktılar



Şekil 1.2. YSA sinir hücresi

**Girdiler:** Girdiler nöronlara gelen verilerdir. Girdiler yapay sinir hücresine bir diğer hücreden gelebileceği gibi direk olarak dış dünyadan da gelebilir. Bu girdilerden gelen veriler biyolojik sinir hücrelerinde olduğu gibi toplanmak üzere nöron çekirdeğine gönderilir.

**Ağırlıklar:** Yapay sinir hücresine gelen bilgiler girdiler üzerinden çekirdeğe ulaşmadan önce geldikleri bağlantıların ağırlığıyla çarpılarak çekirdeğe iletilir. Bu sayede girdilerin üretilecek çıktı üzerindeki etkisi ayarlanabilinmektedir. Bu ağırlıkların değerleri pozitif, negatif veya sıfır olabilir. Ağırlığı sıfır olan girdilerin çıktı üzerinde herhangi bir etkisi olmamaktadır.

**Toplama Fonksiyonu (Birleştirme Fonksiyonu):** Toplama fonksiyonu bir yapay sinir hücresine ağırlıklarla çarpılarak gelen girdileri toplayarak o hücrenin net girdisini hesaplayan bir fonksiyondur.

Bazı durumlarda gelen girdilerin değeri dikkate alınırken bazı durumlarda ise gelen girdilerin sayısı önemli olabilmektedir. Bir problem için en uygun toplama fonksiyonu belirlenirken geliştirilmiş bir yöntem yoktur. Genellikle deneme yanılma yoluyla toplama fonksiyonu belirlenmektedir. Bazen her hücrenin toplama fonksiyonunun aynı olması gerekmez. Bu konulara karar vermek tasarımcıya aittir.

Tablo 1.2. Bazı Toplama Fonksiyonları

|  |  |
| --- | --- |
| Toplam | Ağırlık değerleri girdiler ile çarpılır ve bulunan değerler birbirleriyle toplanarak Net girdi hesaplanır. |
| Çarpım | Ağırlık değerleri girdiler ile çarpılır ve daha sonra bulunan değerler birbirleriyle çarpılarak Net Girdi Hesaplanır. |
| Maksimum  Net = Max(Xi\*Wi) | n adet girdi içinden ağırlıklar girdilerle çarpıldıktan sonra içlerinden en büyüğü Net girdi olarak kabul edilir. |
| Minimum  Net = Min(Xi\*Wi) | n adet girdi içinden ağırlıklar girdilerle çarpıldıktan sonra içlerinden en küçüğü Net girdi olarak kabul edilir. |
| Çoğunluk | n adet girdi içinden girdilerle ağırlıklar çarpıldıktan sonra pozitif ile negatif olanların sayısı bulunur. Büyük olan sayı hücrenin net girdisi olarak kabul edilir. |
| Kumilatif Toplam | Hücreye gelen bilgiler ağırlıklı olarak toplanır. Daha önce hücreye gelen bilgilere yeni hesaplanan girdi değerleri eklenerek hücrenin net girdisi hesaplanır |

**Aktivasyon Fonksiyonu:** Bu fonksiyon hücreye gelen net girdiyi işleyerek hücrenin bu girdiye karşılık üreteceği çıktıyı belirler. Aktivasyon fonksiyonu genellikle doğrusal olmayan bir fonksiyon seçilir. Yapay sinir ağlarının bir özelliği olan “doğrusal olmama” aktivasyon fonksiyonlarının doğrusal olmama özelliğinden gelmektedir. Aktivasyon fonksiyonu seçilirken dikkat edilmesi gereken bir diğer nokta ise fonksiyonun türevinin kolay hesaplanabilir olmasıdır. Geri beslemeli ağlarda aktivasyon fonksiyonunun türevi de kullanıldığı için hesaplamanın yavaşlamaması için türevi kolay hesaplanır bir fonksiyon seçilir. Günümüzde en yaygın olarak kullanılan “Çok katmanlı algılayıcı” modelinde genel olarak aktivasyon fonksiyonu olarak “Sigmoid fonksiyonu” kullanılır.

Tablo 1.3. Aktivasyon Fonksiyonları

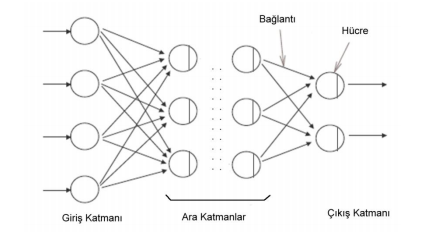
|  |  |  |  |
| --- | --- | --- | --- |
| Doğrusal (Lineer) Aktivasyon Fonksiyonu |  | F(NET)=A\* NET  (A sabit bir sayı) | Doğrusal problemler çözmek amacıyla aktivasyon fonksiyonu doğrusal bir fonksiyon olarak seçilebilir. Toplama fonksiyonundan çıkan sonuç, belli bir katsayı ile çarpılarak hücrenin çıktısı olarak hesaplanır. |
| Adım Aktivasyon Fonksiyonu |  |  | Gelen Net girdinin belirlenen bir eşik değerin altında veya üstünde olmasına göre hücrenin çıktısı 1 veya 0 değerini alır |
| Sigmoid Aktivasyon Fonksiyonu |  |  | Sigmoid aktivasyon fonksiyonu sürekli ve türevi alınabilir bir fonksiyondur. Doğrusal olmayışı dolayısıyla yapay sinir ağı uygulamalarında en sık kullanılan fonksiyondur. Bu fonksiyon girdi değerlerinin her biri için 0 ile 1 arasında bir değer üretir. |
| Tanjant Hiperbolik Aktivasyon Fonksiyonu |  |  | Tanjant hiperbolik fonksiyonu, sigmoid fonksiyonuna benzer bir fonksiyondur. Sigmoid fonksiyonunda çıkış değerleri 0 ile 1 arasında değişirken hiperbolik tanjant fonksiyonunun çıkış değerleri -1 ile 1 arasında değişmektedir |
| Eşik Değer Fonksiyonu |  |  | Gelen bilgilerin 0 dan küçük-eşit olduğunda 0 çıktısı, 1 den büyükeşit olduğunda 1 çıktısı, 0 ile 1 arasında olduğunda ise yine kendisini veren çıktılar üretilebilir. |
| Sinüs Aktivasyon Fonksiyonu |  |  | Öğrenilmesi düşünülen olayların sinüs fonksiyonuna uygun dağılım gösterdiği durumlarda kullanılır. |

**1.6.3 YSA’nın Yapısı**

Yapay sinir ağları yapay sinir hücrelerinin birbirine bağlanmasıyla oluşan yapılardır. Yapay sinir ağları üç ana katmanda incelenir; Giriş Katmanı, Ara (Gizli) Katmanlar ve Çıkış Katmanı.

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

Ara (Gizli) Katman(lar): 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 hücre 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 çıktılarını üreten katmandır. Bu katmanda üretilen çıktılar dış dünyaya gönderilir. Geri beslemeli ağlarda bu katmanda üretilen çıktı kullanılarak ağın yeni ağırlık değerleri hesaplanır.



Şekil 1.3.YSA yapısı2

**1.7. YSA’nın Eğitilmesi**

Eğitilmesi İnsan beyni doğumdan sonraki gelişme sürecinde çevresinden duyu organlarıyla algıladığı davranışları yorumlar ve bu bilgileri diğer davranışlarında kullanır. Yaşadıkça beyin gelişir ve tecrübelenir. Artık olaylar karşısında nasıl tepki göstereceğini çoğu zaman bilmektedir. Fakat hiç karşılaşmadığı bir olay karşısında yine tecrübesiz kalabilir. Yapay sinir ağlarının öğrenme sürecinde de dış ortamdan girişler alınır, aktivasyon fonksiyonundan geçirilerek bir tepki çıkışı üretilir. Bu çıkış yine tecrübeyle verilen çıkışla karşılaştırılarak hata bulunur.

Çeşitli öğrenme algoritmalarıyla hata azaltılıp gerçek çıkışa yaklaşılmaya çalışılır. Bu çalışma süresince yenilenen yapay sinir ağının ağırlıklarıdır. Ağırlıklar her bir çevrimde yenilenerek amaca ulaşılmaya çalışılır. Amaca ulaşmanın veya yaklaşmanın ölçüsü de yine dışarıdan verilen bir değerdir. Eğer yapay sinir ağı verilen giriş-çıkış çiftleriyle amaca ulaşmış ise ağırlık değerleri saklanır.

Ağırlıkların sürekli yenilenerek istenilen sonuca ulaşılana kadar geçen zamana öğrenme adı verilir. Yapay sinir ağı öğrendikten sonra daha önce verilmeyen girişler verilip, sinir ağı çıkışıyla gerçek çıkış yaklaşımı incelenir. Eğer yeni verilen örneklere de doğru yaklaşıyorsa sinir ağı işi öğrenmiş demektir. Sinir ağına verilen örnek sayısı optimum değerden fazla ise sinir ağı işi öğrenmemiş, ezberlemiş demektir. Genelde eldeki örneklerin %80 ağa verilip ağ eğitilir. Daha sonra kalan %20 lik kısım verilip ağın davranışı incelenir. Böylece ağın testi yapılmış olur.

**1. Örneklerin toplanması:** Ağın öğrenmesi istenilen olay için daha önce gerçekleşmiş örneklerin bulunması adımıdır. Ağın eğitilmesi için örnekler toplandığı gibi (eğitim seti) ağın test edilmesi için de örneklerin (test seti) toplanması gerekmektedir. Eğitim setindeki örnekler tek tek gösterilerek ağın olayı öğrenmesi sağlanır. Ağ olayı öğrendikten sonra test setindeki örnekler gösterilerek ağın performansı ölçülür. Hiç görmediği örnekler karşısındaki başarısı ağın iyi öğrenip öğrenmediğini ortaya koyar.

**2. Ağın topolojik yapısının belirlenmesi**: Öğrenilmesi istenen olay için oluşturulacak olan ağın topolojik yapısı belirlenir. Kaç tane girdi ünitesi, kaç tane ara katman, her ara katmanda kaç tane proses eleman kaç tane çıktı eleman olması gerektiği bu adımda belirlenmektedir.

**3. Öğrenme parametrelerinin belirlenmesi:** Ağın öğrenme katsayısı, proses elemanlarının toplama ve aktivasyon fonksiyonları, momentum katsayısı gibi parametreler bu adımda belirlenmektedir.

**4. Ağırlıkların başlangıç değerlerinin atanması:** Proses elemanlarını birbirlerine bağlayan ağırlık değerlerinin ve eşik değer ünitesinin ağırlıklarının başlangıç değerlerinin atanması yapılır. Başlangıç genellikle rasgele değerler atanır. Daha sonra ağ uygun değerleri öğrenme sırasında kendisi belirler.

**5. Öğrenme setinden örneklerin seçilmesi ve ağa gösterilmesi:** Ağın öğrenmeye başlaması ve Öğrenme kuralına uygun olarak ağırlıkları değiştirmesi için ağa örnekler belirli bir düzeneğe göre gösterilir.

**6. Öğrenme sırasında ileri hesaplamaların yapılması:** Sunulan girdi için ağın çıktı değerleri hesaplanır.

**7. Gerçekleşen çıktının beklenen çıktı ile karşılaştırılması:** Ağın ürettiği hata değerleri bu adımda hesaplanır.

**8. Ağırlıkların değiştirilmesi:** Geri hesaplama yöntemi uygulanarak üretilen hatanın azalması için ağırlıkların değiştirilmesi yapılır.

**9.Öğrenmenin tamamlanması:** İleri beslemeli sinir ağı öğrenmeyi tamamlayıncaya, yani gerçekleşen ile beklenen çıktılar arasındaki hatalar kabul edilir düzeye ininceye kadar devam eder.

Ağın kendisine gösterilen girdi örneği için beklenen çıktıyı üretmesini sağlayacak ağırlık değerleri başlangıçta rastgele atanmakta ve ağa örnekler gösterildikçe ağırlıklar değiştirilerek istenen değerlere ulaşması sağlanmaktadır. İstenen ağırlık değerlerinin ne olduğu bilinmemektedir. Bu nedenle YSA nın davranışlarını yorumlamak ve açıklamak mümkün olmaz.

Bazı durumlarda ağın takıldığı yer hata düzeyinin üstünde kalabilir. Bu durumda ağın olayı öğrenmesi için bazı değişiklikler yapılarak yeniden eğitilmesi gerekir. Bunlar; Başlangıç değerlerinde değişiklik yapılabilir. Ağ topolojisinde değişiklikler yapılabilir. Ağın parametrelerinde değişiklikler yapılabilir. Ağa sunulan verilerin gösterimi ve örneklerin formülasyonu değiştirilerek yeni örnek seti oluşturulabilir. Öğrenme setindeki örneklerin sayısı artırılabilir veya azaltılabilir.

İleri beslemeli sinir ağının yerel sonuçlara takılıp kalmaması için momentum katsayısı geliştirilmiştir. Ağların eğitilmesinde diğer önemli bir sorun ise öğrenme süresinin çok uzun olmasıdır. Ağırlık değerleri başlangıçta büyük değerler olması durumunda ağın lokal sonuçlara düşmesi ve bir lokal sonuçtan diğerine sıçramasına sebep olmaktadır. Eğer ağırlıklar küçük aralıkta seçilirse o zaman da ağırlıkların doğru değerleri bulması uzun zamanlar almaktadır. Bazı problemlerin çözümü sadece 200 iterasyon sürerken bazıları 5-10 milyon iterasyon sürmektedir.

Ağın öğrenmesinin gösterilmesinin en güzel yolu hata grafiğini çizmektir. Her iterasyonda oluşan hatanın grafiği çizilirse hatanın zaman içinde düştüğü gözlenebilir. Belirli bir iterasyondan sonra hatanın daha fazla azalmayacağı görülür. Bu ağın öğrenmesinin durduğu ve daha iyi bir sonuç bulunamayacağı anlamına gelir. Eğer elde edilen çözüm kabul edilemez ise o zaman ağ yerel bir çözüme takılmış demektir.

**1.7.1 Katsayıların Belirlenmesi**

İleri beslemeli sinir ağında bağlantıların ağırlık değerlerinin başlangıçta belirlenmesi ağın performansını yakından ilgilendirmektedir. Genel olarak ağırlıklar belirli aralıklarda atanmaktadır. Bu aralık eğer büyük tutulursa yerel çözümler arasında sürekli dolaştığı küçük olması durumunda ise öğrenmenin geç gerçekleştiği görülmüştür. Bu değerlerin atanması için henüz standart bir yöntem yoktur. Tecrübeler 0.1 ile 1.0 arasındaki değerlerin başarılı sonuçlar ürettiğini göstermektedir. Fakat bu tamamen öğrenilmesi istenen problemin niteliğine bağlıdır.

Başlangıç değerleri kadar öğrenme ve momentum katsayılarının belirlenmesi de ağın öğrenme performansı ile yakından ilgilidir. Öğrenme katsayısı ağırlıkların değişim miktarını belirlemektedir. Eğer büyük değerler seçilirse o zaman yerel çözümler arasında ağın dolaşması ve osilasyon yaşaması söz konusu olmaktadır. Küçük değerler seçilmesi ise öğrenme zamanını artırmaktadır. Tecrübeler genellikle 0.2-0.4 arasındaki değerlerin kullanıldığını göstermiştir. Bazı uygulamalar öğrenme katsayısının 0.6 değerini aldığı zaman en başarılı sonuçları verdiği göstermiştir. Bu durum tamamen probleme bağlıdır.

Benzer şekilde momentum katsayısı da öğrenmenin performansını etkiler. Momentum katsayısı bir önceki iterasyondaki değişimin belirli bir oranının yeni değişim miktarına eklenmesidir. Bu özellikle yerel çözümlere takılan ağların sıçrama ile daha iyi sonuçlar bulmasını sağlamak amacıyla ile önerilmiştir. Bu değerlerin küçük olması yerel çözümlerden kurtulmayı zorlaştırabilir. Çok büyük değerler ise çözüme ulaşmada sorunlar yaşatabilir. Tecrübeler momentum katsayısının 0.6-0.8 arasında seçilmesinin uygun olacağını göstermiştir. Problemin niteliğine göre kullanıcının almasında fayda vardır.[3]

**BÖLÜM 2. GENETİK ALGORİTMA**

Geniş çözüm uzaylarının klasik yöntemlerle taranması hesaplama zamanını artırmaktadır. Genetik algoritma ile kabul edilebilir doğrulukta kısa sürede bir sonuca gidilebilir. Genetik algoritmalar özellikle çözüm uzayının geniş, süreksiz, karmaşık olduğu durumlarda başarılı sonuçlar vermektedir.

Genetik algoritma bir arama ve optimizasyon tekniğidir. Genetik algoritma tek bir noktadan değil noktalar kümesinden geniş bir alandan arama yapmaktadır. Olasılık içeren kurallar kullanır. Türev ve benzeri yardımcı bilgileri kullanmaz. Yalnızca amaç fonksiyonu bilgisine ihtiyaç duyar.

Genetik algoritma genel anlamda, dizilerden oluşan bir populasyona (nesil) çoğalma, çaprazlama ve mutasyon operatörlerinin uygulanmasını içerir. Bu operatörlerin uygulanmasından sonra yeni bir popülasyon (yavru nesil) oluşur. Yeni nesil eski ebeveyn nesil ile yer değiştirir. Her dizinin (kromozomun) bir uyum değeri vardır. Yeni nesiller (yeni diziler) bu uyum değerlerin göre seçilirler. Her yeni üretilen nesilde daha uyumlu iyi nesiller üretilmeye çalışılır.

**2.1. Temel Bilgiler GA’nın Uygulanması**

Uygulamadaki ilk adım, ilk popülasyonun oluşturulup, uyum değerinin hesaplanmasıdır. Daha sonra mevcut nesile, temel genetik operatörler (çoğalma, çaprazlama, mutasyon) uygulanır. Her nesil için uyum değeri hesaplanır. Bu durum durdurma kriteri sağlanana kadar devam eder. Adımlar şu şekilde açıklanabilir.

* Arama uzayındaki tüm olası çözümlerden bir grup çözüm dizi olarak kodlanır. Genellikle rastsal bir işlem yapılır ve başlangıç popülasyonu oluşturulur.
* Her bir dizi için uyum değeri hesaplanır. Bulunan uyum değerleri dizilerin çözüm kalitesini gösterir.
* Bir grup dizi belirli bir olasılık değerine göre rastsal olarak seçilir. Seçilen diziler çaprazlama ve mutasyon işlemlerine tabi tutulur.
* Oluşan yeni populasyon eski populasyon ile yer değiştirilir.
* Durdurma kriteri sağlanana dek yukarıdaki işlemlere devam edilir. En uygun olan dizi çözüm olarak seçilir.

**Kodlama**

Çözümlerin kodlanması, probleme özgü bilgilerin genetik algoritmanın kullanacağı şekle çevrilmesidir. Her problem kendine özgü farklı kodlamalara ihtiyaç duyabilir. Tüm problemler için geçerli olabilecek genel bir kodlama tekniği yoktur.

Kodlama türleri 3 grubda incelenebilir.

**İkili Kodlama**: En yaygın kodlama yöntemidir. Problemin olası çözümleri 0 ve 1 sayıları ile kodlanır.

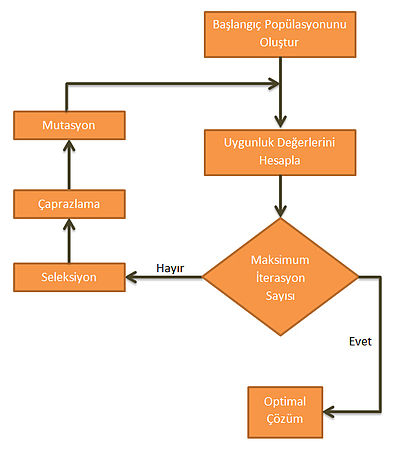
**Sıralı Kodlama:** Genellikle birleşik optimizasyon problemlerinde kullanılmaktadır. Dizinin uyum değeri Gen değerlerine ve genlerin sıralarına da bağlıdır. Gezgin satıcı problemi ve Araç rotalama problemleri bu kodlamaya uygundur.

Gezgin satıcıda her tamsayı değeri bir şehri gösterir. Bu yöntem ile tekrar eden değerler engellenmiş olur. Ayrıca uyum değerlerinin hesaplanması da basit hale gelmiş olur.

**Değer Kodlaması:** Değerler kodlamasında her dizi, bir değerler kümesinden oluşmaktadır. Değerler probleme göre herhangi bir sayı yada karekterler olabilir.

Değer kodlaması bazı özel problemler için oldukça kullanışlıdır. Fakat bu tip kodlama kullanıldığında probleme özgü çaprazlama ve mutasyon yöntemleri gerekir. Böyle bir kodlama yöntemi Yapay Sinir Ağlarında ağırlıkların bulunması amacıyla kullanılabilir. Bu durumda dizide kullanılan gerçek sayılar, yapay sinir ağlarındaki ağırlıkları ifade eder.

**Ağaç Kodlama Yöntemi**: Bu yöntemde her dizi nesnelerden oluşan bir ağaç yapısı şeklinde ifade edilir. Bu nesneler fonksiyonlar yada programlama dillerinde geçen komutlar olabilir. Bu yöntem verilen değerlere uygun bir fonksiyon bulmak için kullanılabilir.



Şekil 2.1. Genetik algoritmanın akış diyağramı.

**Uyum Değeri**

Dizi ile taşınan genetik bilgi her kuşakta uyum değeri daha iyi dizilerin seçilmesine rehberlik eder. Dizinin uyum değeri ne kadar yüksek ise, yaşama ve çoğalma şansıda o kadar yüksek olacaktır.

Problemin türünü göre Uyum değeri farklı şekillerde olur. Amaç fonksiyonu kârı en yüksek tutmak yada, maliyetleri en düşük tutmak olabilir.

**Alt Dizi Kavramı**

Alt diziler genetik algoritmaların davranışlarını açıklamak için kullanılan bir yapıdır. Bir alt dizi belirli dizi kümeleri arasındaki benzerliği tanımlayan bir dizidir. Alt diziler {0,1,\*} harfleri kullanılarak tanımlanır.

Örneğin H=01\*1\* alt dizisi, ilk konumunda 0, ikinci ve dördüncü konumunda 1 değeri olan diziler kümesi içindir. \* sembolü dizinin o konumunun hangi değeri alıp almadığının önemli olmadığını gösterir. Dizi bu konumlarda 0 yada 1 de alabilir. Fakat önemli değildir. Eğer bir dizi alt dizinin örüntüsüne uyarsa bu diziye “H nin bir örneğidir” denir. Alt dizilerin iki özelliği mevcuttur. Bu özellikler aşağıda verilmiştir.

Alt dizi derecesi: Alt dizi örüntüsünde bulunan sabit konumların sayısıdır. Örneğimizde 0 ve 1 değerlerinin sayısının toplamına eşittir. o(H) ile gösterilir.

Alt dizi uzunluğu: Alt dizi örüntüsündeki ilk ve son konumlar arasındaki uzaklığı gösterir. δ(H) ile gösterilir.

Alt dizi derecesi ve uzunluğu genetik algoritmaların temelinde önemli bir yer tutar. Derecesi düşük, uzunluğu kısa olan alt diziler “yapı blokları” olarak adlandırılır. Genetik algoritmaların işleyişinde uygun yapı bloklarının tanımlanmasını ve yapı bloklarının daha uygun yapı blokları elde etmek için birleştirilmesi tavsiye edilir.

“Popülasyon ortalamasının üstünde uyum gücü gösteren, düşük dereceye ve kısa uzunluğa sahip alt diziler zamanla üstsel olarak çoğalırlar. Bu çoğalma sonucu ebeveyn dizilerden daha üstün özelliklere sahip diziler ortaya çıkmaktadır

**2.2. Genetik Algoritma Operatörleri**

Genetik algoritmada kullanılan temel operatörler Çoğalma operatörü (Seçilme Operatörü), Çaprazlama operatörü ve Mutasyon Operatörüdür.

**2.2.1 Çoğalma operatörü (Seçilme operatörü)**

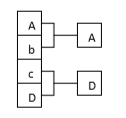
Çoğalma işlemi, dizilerin uyum fonksiyonu (amaç fonksiyonu) değerlerine göre kopyalandığı süreçtir. Yüksek uyum değerine sahip dizilerin bir sonraki kuşağa katkılarının daha yüksek olacağı anlamına gelir. Çoğalma işlemi, dizileri seçme işleminden, seçilmiş dizileri bir eşleme havuzuna kopyalama işleminden, havuzda dizileri genelde çiftler halinde gruplara ayırma işleminden oluşmaktadır. Literatürde bu operatör, seçilme operatörü olarak da geçmektedir. Seçim işlemi bir sonraki kuşak için yavru üretmek amacıyla hangi ailelerin yer alması gerektiğine karar verir. Bu amaçla ortalama uyum değeri üzerindeki dizilere çoğalma fırsatı tanıyarak olur. Literatörde çoğalma yöntemleri Rulet tekerleği (roulette Wheel), turnuva (tournament), sabit durum (steady state) ve sıralama (permutation) yöntemleri sayılabilir.

**Rulet Tekerleği Yöntemi:** Bu yöntemde her dizi uyum değeriyle orantılı bir olasılık değeri ile seçilmektedir. Rulet tekerleğinin yüzeyi dizilerin uyum değerleri ile orantılı olarak işaretlenir. Tekerlek kaç defa döndürülürse her seferinde bir dizi eşleme havuzuna atılır. Daha iyi uyum değerine sahip diziler tekerlekte daha fazla yer aldıklarından onların seçilme şansları daha yüksektir.



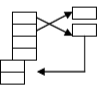
Şekil 2.2. Rulet tekerleği

**Turnuva Yöntemi:** Tüm nesil içinden her seferinde rastgele dizi seçilir. Bunlar turnuvaya katılmış olur. Bu ikisi arasından yapılan turnuvada hangisi daha iyiyse o kazanır. Her kişinin (dizinin) turnavaya katılması iki kez yapılır. Dolayısıyla en iyi kişi (dizi) her iki turnovada başarılı olur. En kötü kişi ise her iki turnuva da başarısız olur. Her iki turnuvada başarılı olan listeye ismini yazdırır. Böylece en başarılı olanlar iki kez orta seviyede olanlar ise bir kez isimlerini yazdırabilirler.



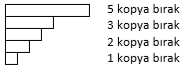
Şekil 2.3.Truva yöntemi

**Sabit Durum Yöntemi:** Bu yöntemde populasyon içindeki sınırlı sayıda dizinin yer değiştirmesini sağlamak, bir dizinin birden fazla kuşak hayatta kalmasını sağlamak için kullanılır. Bu amaçla mevcut populasyon içerisinden belli sayıda dizi seçilir. Bunlar yeni nesiller üretir. Üretilen yeni nesiller tekrar populasyona katılır. Fakat populasyon nüfusunun artmaması için içeriden bazılarının atılması gerekir. Bu nedenle burada karar verilmesi gereken iki işlemden bahsedilebilir. Yeni nesil üretecek dizilerin hangi kriterle seçileceği, Yeni doğan nesillerin ise kimin yerini alacağına nasıl karar verileceğidir.



Şekil 2.4. Sabit Durum yöntemi

**Sıralama Yöntemi:** Bu yöntemde tüm diziler uyum değerlerine göre artan bir şekilde sıralanır. Uygun bir fonksiyonla en yüksek uyumlular havuza daha fazla kopya bırakır, en kötü diziler ise kopya elde edememiş olur. Bu yöntemde arama hızı yüksektir.



Şekil 2.5. Sıralma yöntemi

**2.2.2. Çaprazlama operatörü(Crossover)**

Mevcut gen havuzunun potansiyelini artırmak üzere, bir önceki kuşaktan daha iyi nitelikleri yeni diziler oluşturmak için çaprazlama operatörü kullanılır. Çoğalma işlemi sonucunda elde edilen yeni populasyondan rastgele iki dizi seçilmekte ve karşılıklı çaprazlama işlemine tabi tutulmaktadır.

Çaprazlama işlemi seçilen iki dizinin ortasından rastgele alınan bir noktadan karşılıklı olarak çaprazlama yapılarak gerçekleştirilir. Uygun bir çaprazlama noktası seçildiğinde ebeveynlerden uyum değeri daha yüksek diziler elde edilebilmektedir. Fakat uygun çaprazlama noktası genellikle bilinmediğinden rastsal bir nokta seçilmektedir.

Temel çaprazlama yöntemleri tek noktalı ve çift noktalı çaprazlama yöntemleridir.

Tek noktalı çaprazlama yöntemi: Havuzdaki diziler rastgele eşleşirler. Seçilen her dizi çifti için ilk ve son gen dışında aradaki genlerden rastgele bir yer seçilir. Burası çaprazlama noktasını gösterir. Bu noktadan sonra gelen genler her iki dizide karşılıklı olarak yer değiştirir. Bu işlem için diziler aynı uzunlukta olmalıdır.

Aşağıdaki dizilerde 3. konum çaprazlama noktası olsun. Bu noktadan sonraki genler çaprazlandığında yeni diziler elde edilir.

P1 = 1,0,0,[1,1,1,1,0] => P1’ =1,0,0,[1,0,0,1,0]

P2 = 1,0,1,[1,0,0,1,0] => P2’ =1,0,1,[1,1,1,1,0]

**Çift noktalı çaprazlama yöntemi:** Bu yöntemde dizi üzerinde ilk ve son genler hariç iki tane rastgele nokta seçilir. Çaprazlama işlemi seçilen bu iki nokta arasındaki genlerin yer değişimidir. Ve bu noktalar arasında genler değiştirilir.

Aşağıdaki dizilerde 1. ve 5. genden sonra gelen aralık çaprazlama noktalarımız olsun. Bu iki nokta arasında kalan genleri çaprazlayalım.

P1 = 1,[0,0,1,1],1,1,0 => P1’ =1,[0,1,1,0],1,1,0

P2 = 1,[0,1,1,0],0,1,0 => P2’ =1,[0,0,1,1],0,1,0

**PMX Çaprazlama Yöntemi (Sıra temelli):** Bu yöntem gezgin satıcı ve araç rotalama problemlerinde yaygın olarak kullanılmaktadır. Çünkü bu problemlerde dizileri oluşturan genlerin aynı dizi içinde tekrar etmemesi gerekir. Bu yöntem bu tekrarı engelleyen bir yöntemdir. Ayrıca bu yöntemde seçilen bir alt dizinin korunması sağlanmaktadır. Bu alt dizi çocuk diziye aktarılmaktadır. Bu yöntemde iki ebeveyn diziden yalnız bir adet yavru dizi elde edilmektedir. Yöntem şu şekildedir.

Havuzdan iki ebeveyn dizi rastgele seçilsin. Bunlar P1 ve P2 olsun.

P1=1,2,4,6,3,7,5,8 => dizisinden 4,6,3 nolu alt dizisi ratgele seçilsin.

P2=5,4,1,7,2,6,8,3

C= \_,\_,4,6,3,\_,\_,\_ yavru dizi olur. ikinci diziden bu sayıların olduğu yerleri çıkaralım.

P2= 5,\*,1,7,2,\*,8,\* olur. Bu dizi içerisindeki 1,7,2 sayıları 4,6,3 alt dizisine karşılık gelen konumdadır. 1,7,2 dizisi \* ile gösterilen konumlara Yavru dizi içerisine yerleştirelim. Bu durumda yavru dizi şöyle olur.

C= \_,1,4,6,3,7,\_,2 olur. İkinci dizideki 5 ve 8 sayıları aynı konumlarına çocuk dizide yerine yazılırsa en son çocuk dizi şu hali alır.

C= 5,1,4,6,3,7,8,2

Böylece iki ebeveyn diziden bir çocuk dizi üretilmiş olur.

**OX Çaprazlama (Sıra temelli):** bu yöntemde yine dizideki genlerin tekrarlı olmasını engellemektedir. Sıra temelli yapıya sahiptir. PMX den farklı olarak burada iki ebeveyn diziden yine 2 tane çocuk dizi oluşturulmaktadır. PMX de olduğu gibi yine ebeveyn dizilerden seçilen alt diziler korunarak çocuk dizilere aktarılmaktadır.

Yöntem şu şekildedir. İki tane ebeyn dizi alalım.

P1= 1,3,5,7,9,2,4,6,8 => dizisinden [7,9,2] alt dizisini seçelim. Bu dizinin aynı konumuna ikinci diziden karşılık gelen alt diziyide bulalım.

P2= 7,5,9,2,6,3,1,4,8 => [2,6,3] dizisi yukarıdaki alt diziyle aynı konuma karşılık gelir.

Bu iki alt dizi çocuk dizilere aynı konumda olacak şekilde aktarılırlar.

C1= \_,\_,\_,7,9,2,\_,\_,\_

C2= \_,\_,\_,2,6,3,\_,\_,\_ çocuk diziler bu hale gelir.

Ebeveyn diziler, ikinci kesim noktasından (alt diziden sonra başlayan kısım) başlayarak dizi tekrar bir daha sıralanır.

P1’=4,6,8,1,3,5,7,9,2

P2’ =1,4,8,7,5,9,2,6,3

Yeni sıralanan birinci ebeveynden ikincinin alt dizisi çıkarılır. Aynı şekilde ikinci ebeveynden de birincinin alt dizisi çıkarılır.

P1’’=4,\*,8,1,\*,5,7,9,\*

P2’’=1,4,8,\*,5,\*,\*,6,3

Çıkarılan bu konumlar olmadan dizi düzenlenir.

P1’’’=4,8,1,5,7,9

P2’’’=1,4,8,5,6,3

Elde edilen bu yeni kısa diziler P1’’’ olan C2 cocuk dizisinin ikinci kesim noktasından sonra (alt dizi bitiminden) tekrar sırayla yerleştirilirAynı şekilde P2’’’ elemanları da C1 cocuk dizisine yerleştirilir.

C1= 5,6,3,7,9,2,1,4,8

C2= 5,7,9,2,6,3,4,8,1

Böylece içerisinde tekrarlı sayıların bulunmadığı iki yeni çocuk dizi elde edilmiş olur.

**CX Çaprazlama (Cycle Crossover)(Sıra temelli):** Bu yöntemde diğer iki yöntem gibi sıra temelli (dizide tekrarlı gen yoktur) yöntemdir. Bu yöntemde her seçilen ebeveyn çiftenden iki tane yavru dizi üretilir. Ebeveyn dizilerde çaprazlama noktası seçilmez. Yöntem şu şekildedir.

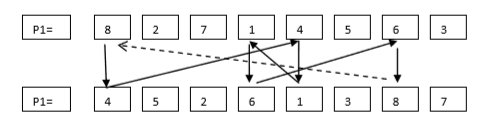
Ebeveyn dizilerimiz şu şekilde olsun.

P1 =8,2,7,1,4,5,6,3

P2 =4,5,6,2,1,3,8,7

İlk ebeveyn dizinin en sol konumundan işleme başlanır. Her bir diziden seçilen genler çocuk dizilere yerleştirilir. İlk gen 8 dir. Bu sayı ile aynı konumda diğer dizide 4 vardır. 4 sayısı birinci dizide beşinci konumdadır.

Aynı konumda ikinci dizide 1 vardır. 1 sayısı birinci dizide dördüncü konumdadır. Aynı konumda ikinci dizide 6 vardır. 6 sayısı birinci dizide yedinci konumdadır. Aynı konumda ikinci dizide 8 vardır. Fakat 8 sayısı daha önce yerleştirildiği için birinci diziden seçim yapılamaz. Bulunan bu ilk genler çocuk dizilere yerleştirilir.



Şekil2.6.CX çaprazlama

Buna göre çocuk diziler şu şekilde olur.

C1 =8,\*,\*,1,4,\*,6,\*

C2 =4,\*,\*,6,1,\*,8,\*

Kalan \* yıldızlı kısımlar ise her iki dizide karşılıklı olarak değiştirilir.

C1 =8,5,2,1,4,3,6,7

C2 =4,2,7,6,1,5,8,3

Böylece iki adet ebeveynden yeni iki adet cocuk dizi üretilmiş olur.

**LOX Çaprazlama (Lineer Order Crossover):** İş çizelgeleme problemlerinde kullanılan bir yöntemdir. Diziler iki noktadan bölgelere ayrılır. Uygun bir şekilde çaprazlanarak yeni nesiller üretilir.

P1= 6 3 1 4 5 8 2 7

P1= 8 7 6 1 3 4 5 2

**Ağaç Çaprazlama:** Alt kısımları bulunan nesnel yapı şeklindeki dizileri çaprazlarken kullanılır. Bir ağacın (dizinin) kesilen alt dalları başka bir ağacın dallarına monte edilir.

**2.2.3 Mutasyon operatörü (Mutation)**

Çaprazlama işlemi, mevcut gen potansiyelini araştırır. Eğer populasyon problemi çözmek için gereksinim duyulan tüm şifrelenmiş bilgiyi içermezse tatmin edici bir çözüme ulaşılamaz. Bu nedenle mevcut gen havuzundan yeni diziler üretme yeteneğine sahip bir operatöre ihtiyaç duyulur. Bu görevi mutasyon gerçekleştirir. Böylece genetik çeşitlilik artırılmış olur. Mutasyon problemin herhangi bir yerel optimum noktaya yakınsamasını engelleyebilmektedir. Mutasyon oranları genellikle düşük tutulmaktadır. Bunun nedeni çaprazlama sonucu elde edilen uyum değeri yüksek dizileri kaybetmemektir.

**Değer Değiştirme Yöntemi:** Basit genetik algoritmada dizideki herhangi bir konumun değerinin 1 iken 0 yada tersini yaparak gerçekleştirilir.

P1= 1,0,0,1,1,1,1,0

P1’=1,0,0,1,1,0,1,0

**Kaydırma Yöntemi:** Dizi içerisinde rastsal olarak belirlenen bir blok genin yine aynı dizi içerisinde rastgele bir konuma yerleştirilmesidir.

P1 = AKDEMZIF

P1’= AEMZIKDF

**Yerleştirme Yöntemi:** Dizi içinde rastgele bir bir genin seçilerek, yine aynı dizi içinde rastgele başka bir konuma yerleştirilmesidir.

P1 = AKDEMZIF

P1’= ADEMZKIF

**Karşılıklı Değişim Yöntemi:** Dizi içerisinde Rastsal olarak iki gen seçilmekte ve bu iki genler birbiriyle yer değiştirmektedir.

P1 = AKDEMZIF

P1’= AZDEMKIF

**2.2.4. Durdurma kriteri**

Çoğalma, çaprazlama ve mutasyon işlemlerinden sonra yeni bir kuşak oluşmaktadır. Tüm bu işlemler sonsuz döngü içerisinde yapılır. Eğer bir durdurma kriteri belirlenmez ise bu süreç sonsuza kadar devam eder. Bununla ilgili olarak şu yöntemler kullanılmaktadır.

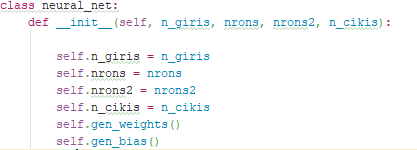
**Hesaplama Zamanı Kriteri:** Bu yöntemde önceden bir hesaplama zamanı veya döngü sayısı belirlenmekte, bu zaman veya döngü sayısına ulaşıldığında durdurulmaktadır. Bu yöntemin sakıncaları olabilmektedir. Döngü devam ettirildiğinde iyileşmeler görülebilecektir. Belirlenen döngü sayısı gerektiğinden de fazla olabilir.

**Optimizasyon Hedefi Kriteri:** Önceden ulaşılması istenen amaç fonksiyonu değeri bilinmektedir. Uyum değeri bu değere ulaştığında algoritma durdurulmaktadır.

Minimum iyileşme kriteri: Genetik algoritma problemlerinde bulunan en iyi çözümler önce hızlı daha sonra yavaş yavaş artış göstermektedir. Bulunan değerlerdeki iyileşme hızının giderek azalması ve sıfıra yaklaşması, artık daha fazla iyileşme beklenmemesi gerektiğini gösterebilir. Çözüme harcanacak zaman ile çözümden beklenecek kalite arasında bir denge kurularak durdurulur.[1]

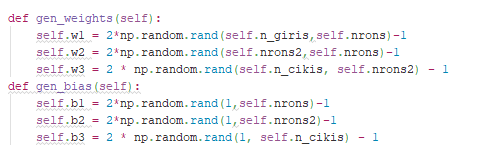
**BÖLÜM 3. YSA VE GA’NIN GOOGLE DİNOZOR OYUNUNA ENTEGRE EDİLMESİ**

**3.1 Yapay Sinir Ağı**



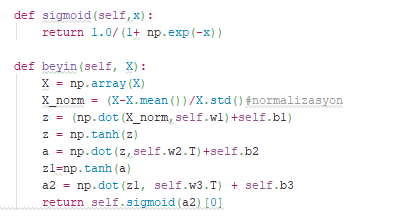
Şekil3.1 YSA yapıcı method

İlk olarak yapay sinir ağının yapıcı metodu oluşturuldu.Burada bir giriş katmanı iki tane gizili katman ve bir çıkış katmanı bulunmaktadır.



Şekil3.2 weight ve bias

Weight ve biaslara random değerler atandı ve aynı zamanda katmanların arasındaki bağlantılar kurulmuş oldu.[4]



Şekil3.3 Aktivasyon fonksiyonu

Beyin fonksiyonunda ilk olarak normalizayon yapılıyor. Giriş değerleri ile weightle çarpılıp biasla toplandıktan sonra tanh fonksiyonuna sokuluyor.Burdan gelen değerler ikinci weight ve biasla çarpılıp toplandıktan sonra tekrar tanh fonksiyonuna sokuluyor.Elde edilen değerler üçüncü weight ve biasla çarpılıp toplandıktan sonra sigmoid fonksiyonuna sokuluyor ve bu değer döndürülüyor.Sigmoid fonksiyonunun kullanılma nedeni iki çıkışın olmasıdır.(zıplama ve eğilme)

**3.1.1 YSA girdileri ve çıktıları**

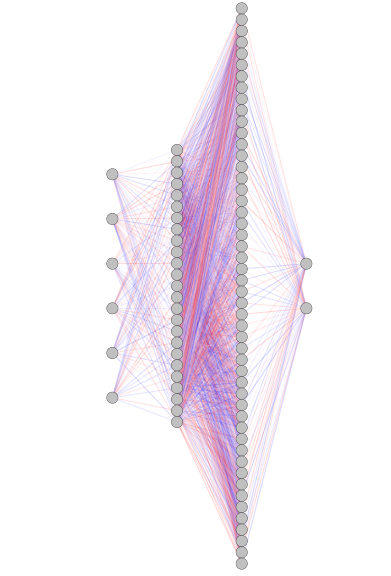
Dinozorların engel görüğünde zamanında eğilmesi veya zıplayabilmesi için yapay sinir ağı bir takım girdilerin verilmesi gerekir.

**Girdiler**

* İlk engelin uzaklığı
* İkinci engelein uzaklığı
* Engelin alt konumu
* Engelin genişliği
* Engelin yüksekliği
* Hız

**Çıktılar**

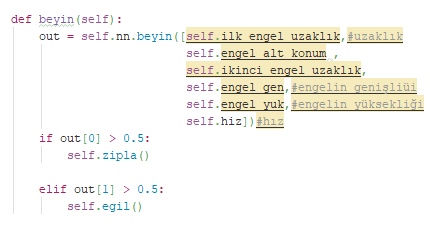
* Zıplama
* Eğilme



Şekil3.4 Temsili YSA Model[2]



Şekil3.5 YSA’da kullanılacak katmanlardaki nöron sayısının belirlenmesi.



Şekil3.6 Dinozorun eğilme ve zıplaması

Burada girdiler YSA’ya gönderiliyor ve çıkış elde ediliyor. Çıkış sigmoid fonksiyonuna gönderildiği için yalnıza sıfır veya bir değeri elde ediliyor. Dinozor çıkıştaki değere göre eğiliyor veya zıplayor.



Şekil3.7Engel bilgileri

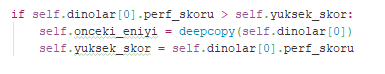
Burada ekranda gözüken engellerin konumları genişleikleri vb. giriş bilgileri atanıyor.

**3.2.Genetik Algoritma**



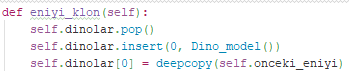
Şekil3.8 Dinozorları skoruna göre sralama

Skora göre dinozorları sıralama.

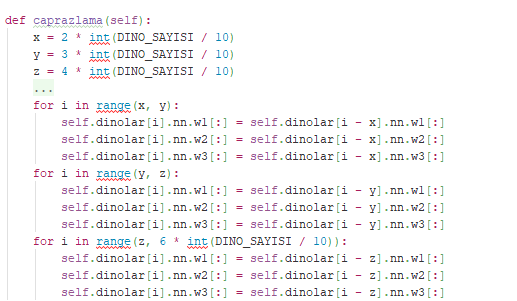


Şekil3.9 Yüsek skorun değişip değişmeyeceğini belirleyen kod

Mevcut jenerasyondaki en yüksek skor,o zamana kadarki en yüksek skordan büyükse mevcut yüksek skor en yüksek skor yapılır.

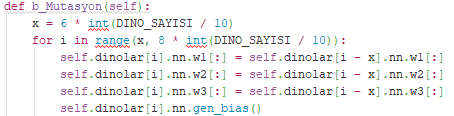


Şekil 3.10En iyi klonun atanması



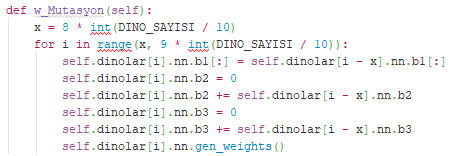
Şekil3.11 Çaprazlama metodu

20.dinozordan 30. Dinozora kadar sırasıyla ilk on dinozorun ağırlıkları atanır aynı şey 30 dan 40 a kadar içinde gerçekleştirilir.40 tan 60 a kadar da ilk 20 dinozorun ağırlıkları sırasıyla atanır.



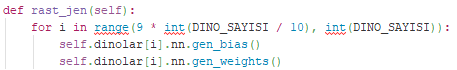
Şekil3.12 Bias mutasyonlama metodu

60 80 arasında weightler ilk 10 dinozorundan kalıtım alıp random bias değerleri atanır



Şekil3.13 Weight mutasyonlama metodu

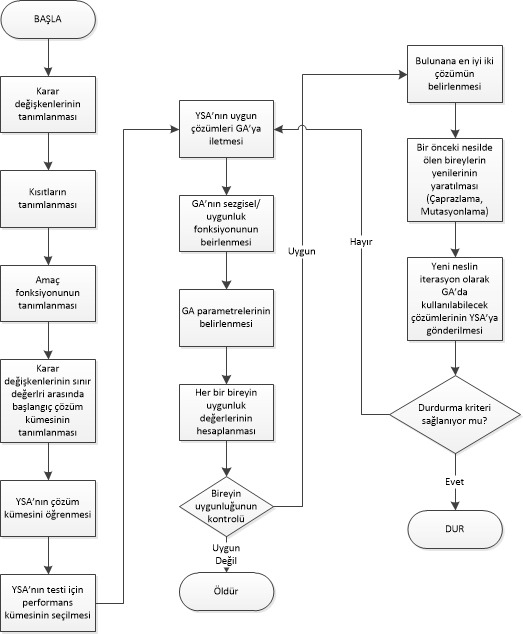
80 90 arasında biaslar ilk 10 dinozordan kalıtım alıp random weight değerleri atanır



Şekil 3.14 Weight ve biaslara random değer atama

En kötü performans gösteren 10 dinozorun weight ve bias değerleri tamamen random değerlerle değiştirilir

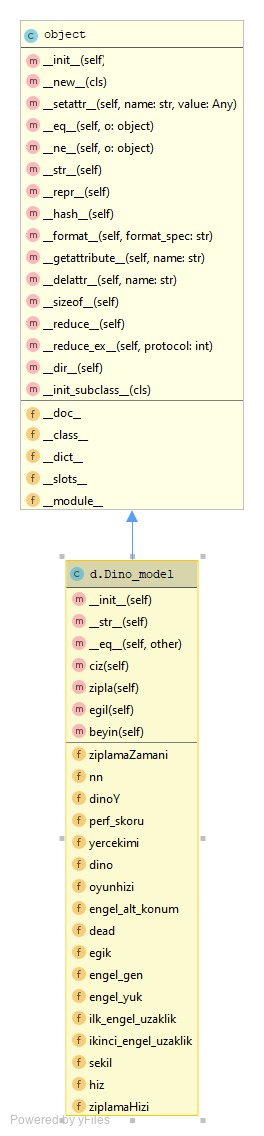
**3.3. Yöntemin Detaylı İşleyişi**

****

Şekil3.15. Yöntemin detaylı işleyişi

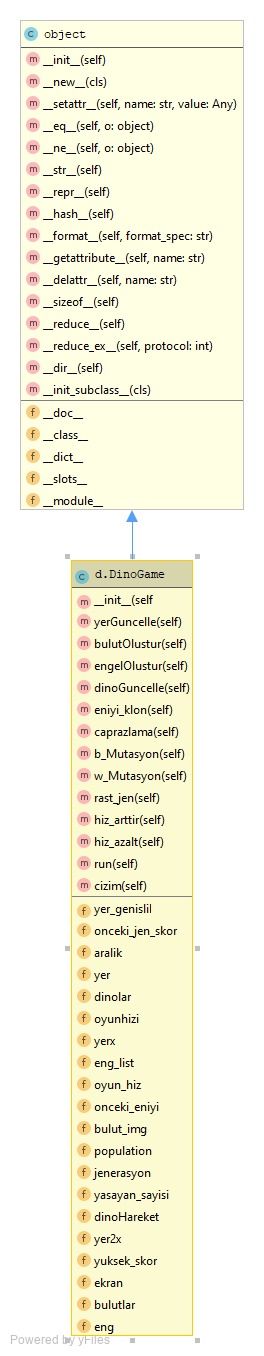
**3.4. UML Diagramı**

**3.4.1. Dino model UML diagramı**



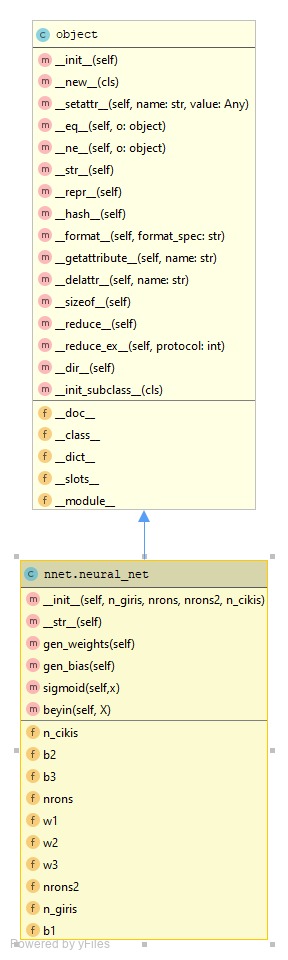
Şekil3.16. Dino model Uml

**3.4.2. Dino game UML diagramı**



Şekil3.17. Dino game Uml

**3.4.3. YSA UML diagramı**



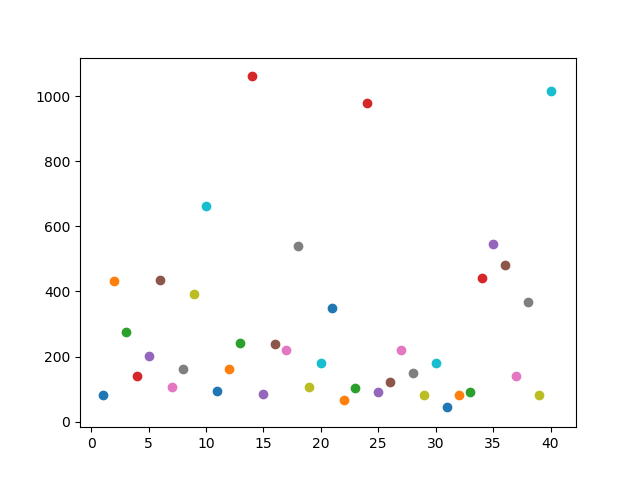
Şekil3.18. YSA Uml

**BÖLÜM4. SONUÇLAR**

Başlangıçta elimizde yapay sinir ağı ve genetik algoritması olmayan bir dinozor oyunu vardı. İlk olarak YSA da kullanılacak parametreleri belirdik. Daha sonra hangi aktivasyon fonksiyonunun projemiz için daha iyi olacağına karar verdik.

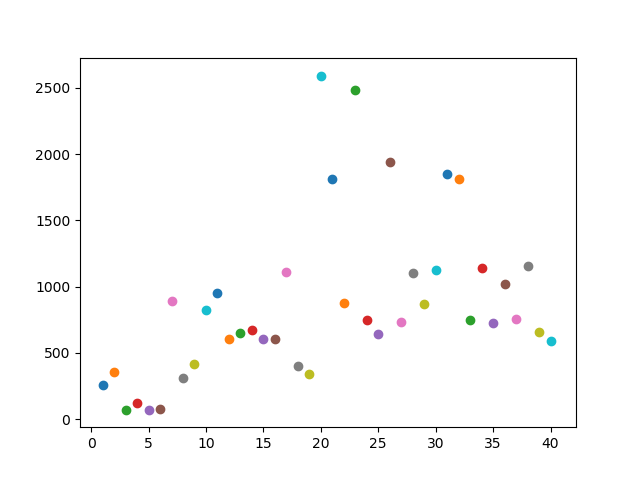
YSA’yı dinozor oyununa entegre ettikten sonra dinozorlar rastgele bir şekilde engelleri eğilerek, zıplayarak yada hiç birşey yapmayarak geçmeye çalışıyordu. Ancak önceki jenerasyon bir sonraki jenerasyona öğrendiği hiç bir şeyi aktarmıyordu.Bu kısımdan itibaren genetik algoritmayı eklemeye başladık. Genleri mutasyona uğratmada sorun yaşamasakta çaprazlamada kullanabileceğimiz birden fazla yöntem vardı.Bir nokta seçerek gen çaprazlamayı denedik ama istediğimiz sonuca tam olarak ulaşamadık.Bizde dinozorları skoruna göre sıralayıp en iyi performans gösteren ilk 10 dinozorun genleri sonraki nesile aktardık.Buna ragmen dinozorlar çok yavaş öğreniyorlardı.Katmanlardaki nöron sayısını ve gizli katman sayısını değiştirdik. Bu işelemler sonucunda ulaşabileceğimiz en optimum çözüme ulaştık.

**4.1. Yaptığımız Denemeler**



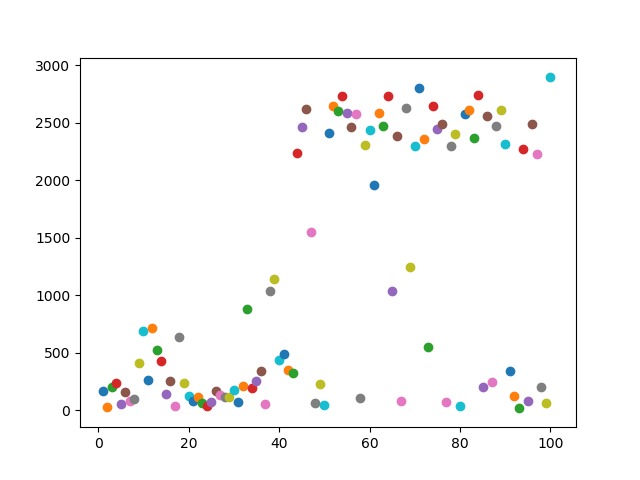
Şekil 4.1 50 ve 100 lük 2 gizli katman 40 tekrar GA yok

Bu grafikte genetik algoritma kullanılmadan 50 ve 100lük 2 gizli katmanla 40 jenerasyonda dinozorların aldığı skorlar görülmektedir.



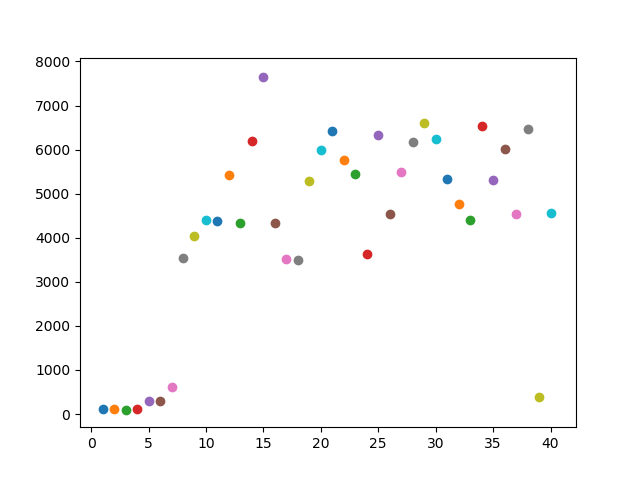
Şekil4.2 30 luk tek gizli katman GA var

Bu grafikte genetik algoritma kullanılırken 30luk tek gizli katmanla 40 jenerasyonda dinozorların aldığı skorlar görülmektedir.



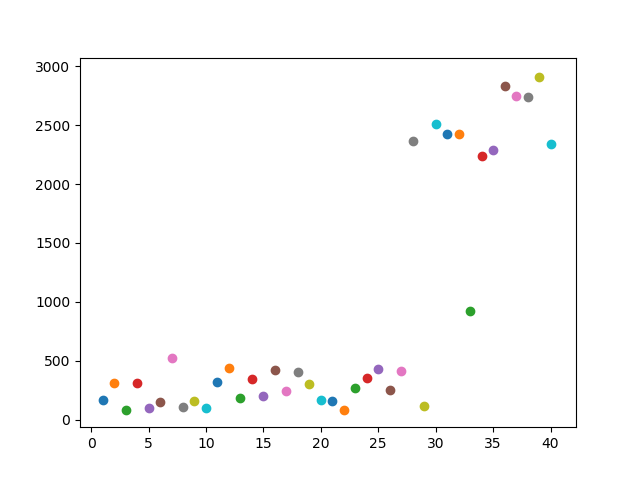
Şekil 4.3 30 luk tek gizli katman 100 jenerasyon GA var

Bu grafikte genetik algoritma kullanılırken 30luk tek gizli katmanla 100 jenerasyonda dinozorların aldığı skorlar görülmektedir.Görüldüğü üzere tekrar sayısı arttıkça algoritma daha iyi öğrenmekte ama bu çok yavaş olduğu için istenilen değildir.



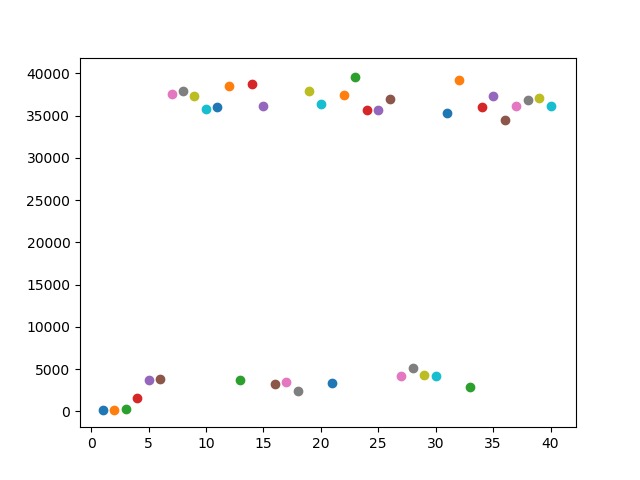
Şekil 4.4 50 ve 100 lük 2 gizli katman 40 jenarasyon

Bu grafikte genetik algoritma kullanılırken 50 ve 100lük 2 gizli katmanla 40 jenerasyonda dinozorların aldığı skorlar görülmektedir.



Şekil 4.5 75 ve 125 lik 2 gizli katman 40 jenerasyon oyun hızı 13

Bu grafikte genetik algoritma kullanılırken 75 ve125lik 2 gizli katmanla 40 jenerasyonda dinozorların aldığı skorlar görülmektedir.Ayrıca burada oyun hızı diğer denemelere göre (diğerlerinde 7 bunda 13)daha fazladır.

100 Şekil 4.6 100ve 150 lik 2 gizli katman 40 jenerasyon oyun hızı 5

Bu grafikte genetik algoritma kullanılırken 100 ve 150lik 2 gizli katmanla 40 jenerasyonda dinozorların aldığı skorlar görülmektedir.Ayrıca burada oyun hızı diğer denemelere göre (diğerlerinde 7 bunda 5)daha azdır.En yüksek skorları bu denemede yakalamış olup oyun hızı ne kadar az ve tekrar sayısı ne kadar çok olursa jenerasyonların başarısının o kadar yüksek olduğu gözlemlendi.

**KAYNAKLAR**

|  |  |  |
| --- | --- | --- |
| [1] |  | www.ibrahimcayiroglu.com. |
| [2] |  | http://alexlenail.me/NN-SVG/index.html |
| [3] |  | https://github.com/fchollet/deep-learning-with-python-notebooks |
| [4] |  | https://www.kaggle.com/kanncaa1/deep-learning-tutorial-for-beginners |

**ÖZGEÇMİŞ**

Levent Can Şençamlar, 26.11.1996 de İzmir’de doğdu.İlkokulunu Fevzi Çakmak İlköğretim Okulunda okudu.Liseyi Övgü Terzibaşıoğlu Anadolu Lisesinde 2014 te tamamladı.2014 yılında Sakarya Üniversitesi Bilgisayar Mühendisliği Bölümü’nü kazandı. 2017 yılında OTOKAR firmasında yazılım stajını yaptı.SAÜ Bilgisayar Mühendisliği Bölümünden 2019 yılında mezun olmayı planlamaktadır.

Yasemin Çerçi, 15.10.1995 de Bursa’da doğdu.İlkokulunu Faik Yılmaz İpek ve Namazgah İhsan Dikmen İlköğretim Okullarında okudu.Liseyi Kırcılar Ticaret meslek Lisesinde okumaya başladı ilk yılın sonunda Tophane Teknik Lisesine Geçiş yaptı ve 2013 te bilgisayar bölümünden mezun oldu.2014 yılında Sakarya Üniversitesi Bilgisayar Mühendisliği MTOK Bölümü’nü kazandı. 2017 yılında OTOKAR firmasında yazılım stajını yaptı.SAÜ Bilgisayar Mühendisliği Bölümünden 2019 yılında mezun olmayı planlamaktadır.

**BSM 498 BİTİRME ÇALIŞMASI**

**Değerlendİrme ve Sözlü Sınav Tutanağı**

KONU : YAPAY ZEKÂ İLE GOOGLE DİNOZOR OYUNU PROGRAMLAMA

ÖĞRENCİLER (Öğrenci No/AD/SOYAD):B140910053 Yasemin ÇERÇİ

B141210026 Levent Can ŞENÇAMLAR

|  |  |  |  |
| --- | --- | --- | --- |
| Değerlendirme Konusu | İstenenler | Not Aralığı | Not |
| **Yazılı Çalışma** |  |  |  |
| **Çalışma klavuza uygun olarak hazırlanmış mı?** | x | 0-5 |  |
| **Teknik Yönden** |  |  |  |
| **Problemin tanımı yapılmış mı?** | x | 0-5 |  |
| Geliştirilecek yazılımın/donanımın mimarisini içeren blok şeması (yazılımlar için veri akış şeması (dfd) da olabilir) çizilerek açıklanmış mı? |  |  |  |
| Blok şemadaki birimler arasındaki bilgi akışına ait model/gösterim var mı? |  |  |  |
| Yazılımın gereksinim listesi oluşturulmuş mu? |  |  |  |
| Kullanılan/kullanılması düşünülen araçlar/teknolojiler anlatılmış mı? |  |  |  |
| Donanımların programlanması/konfigürasyonu için yazılım gereksinimleri belirtilmiş mi? |  |  |  |
| UML ile modelleme yapılmış mı? |  |  |  |
| Veritabanları kullanılmış ise kavramsal model çıkarılmış mı? (Varlık ilişki modeli, noSQL kavramsal modelleri v.b.) |  |  |  |
| Projeye yönelik iş-zaman çizelgesi çıkarılarak maliyet analizi yapılmış mı? |  |  |  |
| Donanım bileşenlerinin maliyet analizi (prototip-adetli seri üretim vb.) çıkarılmış mı? |  |  |  |
| Donanım için gerekli enerji analizi (minimum-uyku-aktif-maksimum) yapılmış mı? |  |  |  |
| Grup çalışmalarında grup üyelerinin görev tanımları verilmiş mi (iş-zaman çizelgesinde belirtilebilir)? |  |  |  |
| Sürüm denetim sistemi (Version Control System; Git, Subversion v.s.) kullanılmış mı? |  |  |  |
| Sistemin genel testi için uygulanan metotlar ve iyileştirme süreçlerinin dökümü verilmiş mi? |  |  |  |
| Yazılımın sızma testi yapılmış mı? |  |  |  |
| Performans testi yapılmış mı? |  |  |  |
| Tasarımın uygulamasında ortaya çıkan uyumsuzluklar ve aksaklıklar belirtilerek çözüm yöntemleri tartışılmış mı? |  |  |  |
| **Yapılan işlerin zorluk derecesi?** | x | 0-25 |  |
| **Sözlü Sınav** |  |  |  |
| **Yapılan sunum başarılı mı?** | x | 0-5 |  |
| **Soruları yanıtlama yetkinliği?** | x | 0-20 |  |
| **Devam Durumu** |  |  |  |
| **Öğrenci dönem içerisindeki raporlarını düzenli olarak hazırladı mı?** | x | 0-5 |  |
| **Diğer Maddeler** |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
| **Toplam** |  |  |  |

Danışman (Jüri adına): Dr.Öğr.Üyesi Seçkin ARI

danışman imzası: