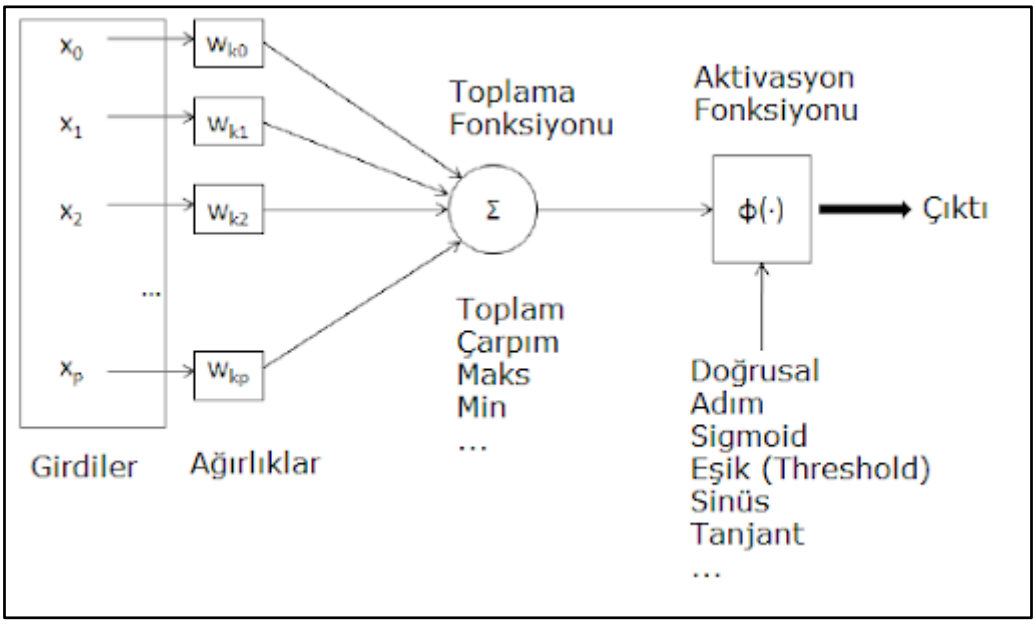
Uygulama 4. Yapay Sinir Ağlarının Geliştirilmesi ve Kullanılması

Canlıların davranışlarını inceleyip, matematiksel olarak modelleyip, benzer yapay modellerin üretilmesine ***sibernetik*** denir. Eğitilebilir, adaptif ve kendi kendine organize olup öğrenebilen ve değerlendirme yapabilen yapay sinir ağları ile insan beyninin öğrenme yapısı modellenmeye çalışılmaktadır. Aynı insanda olduğu gibi yapay sinir ağları vasıtasıyla makinelerin eğitilmesi, öğrenmesi ve karar vermesi amaçlanmaktadır. Şekil 1'de biyolojik sinir sistemi ile yapay sinir sistemi karşılaştırılmıştır.

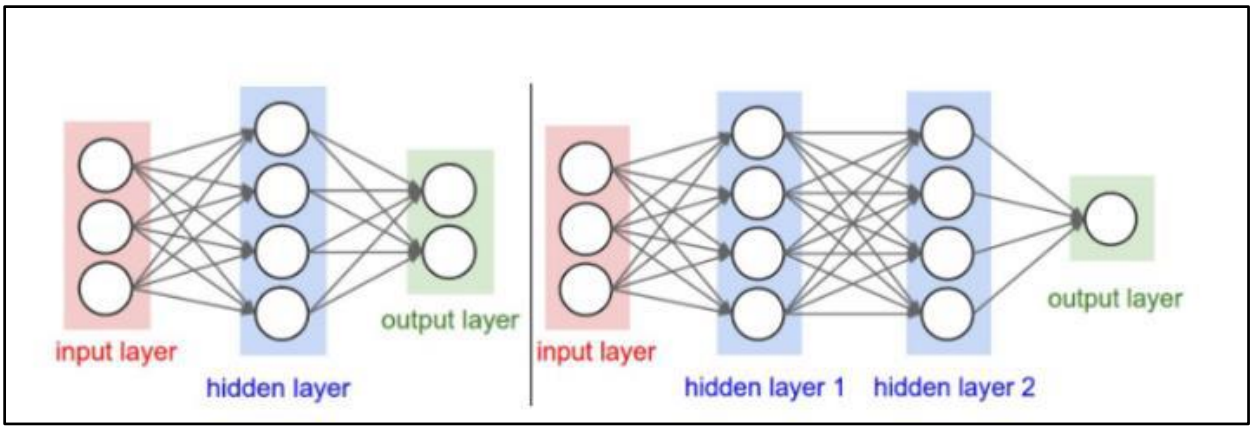
Biyolojik Sinir Sistemi	**Yapay Sinir Sistemi**
Nöron	İşlemci eleman
Dentrit	Toplama fonksiyonu
Hücre gövdesi	Transfer fonksiyonu
Aksonlar	Yapay nöron çıkışı
Sinapslar	Ağırlıklar

Şekil 1. Sinir sisteminin biyoloji ve yapay ağlardaki gösterimi.



Şekil 2: Yapay sinir ağının genel yapısı

Yapay sinir ağları tek katmanlı ve çok katmanlı olmak üzere ayrı ayrı ifade edilebilirler. Şekil 3'te tek ve çok katmanlı yapay sinir ağına örnekler verilmiştir.



Şekil 3. Tek ve çok katmanlı yapay sinir ağı.

Girdi katmanı: Bu katman, girdileri almaktan ve diğer katmanlara iletmekten başka hiçbir şey yapmayan nöronlardan oluşur. Giriş katmanındaki katman sayısı, veri kümesindeki niteliklere veya özelliklere eşit olmalıdır.

Gizli Katman: Giriş ve çıkış katmanı arasında, modelin türüne göre gizli katmanlar olacaktır. Gizli katmanlar çok sayıda nöron içerir. Gizli katmandaki nöronlar, girdilere ve bunları geçmeden önce dönüşümleri uygular. Ağ eğitildiği için ağırlıklar güncellenir, daha öngörücüdür.

Toplama Fonksiyonu (Birleştirme Fonksiyonu): Toplama fonksiyonu bir yapay sinir hücresine ağırlıklarla çarpılarak gelen girdileri toplayarak o hücrenin net girdisini hesaplayan bir fonksiyondur. Tablo 1'de kullanılan bazı toplama fonksiyonları verilmiştir.

Tablo 1. Bazı toplama fonksiyonları

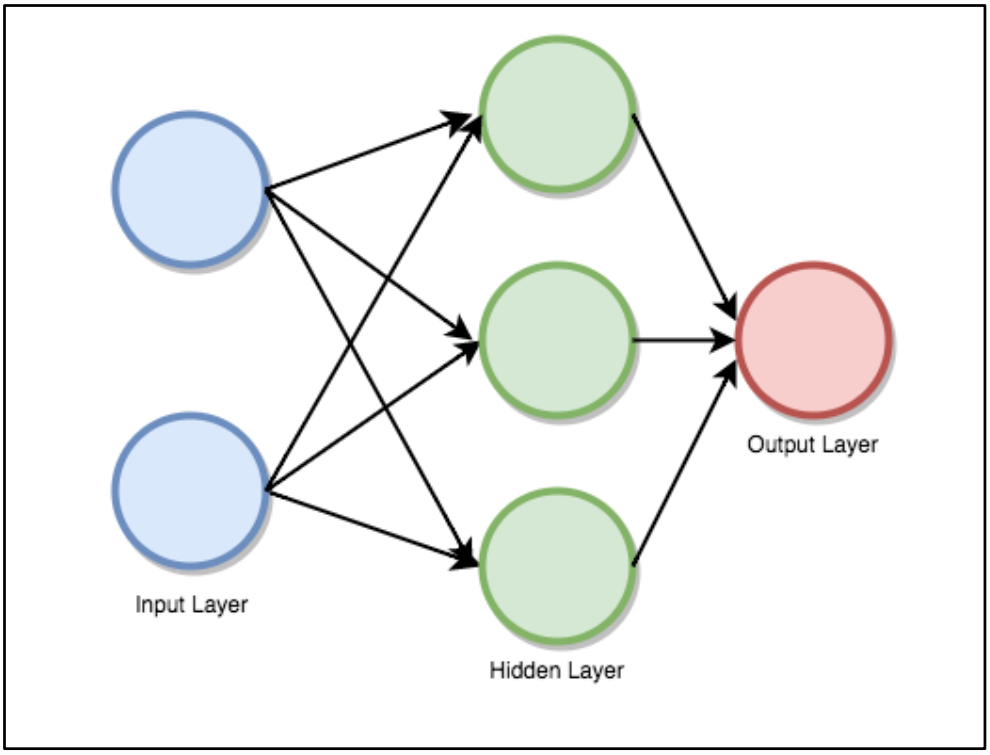
Toplam $\text{Net} = \sum_{i=1}^{N} X_i * W_i$	Ağırlık değerleri girdiler ile çarpılır ve bulunan değerler birbirleriyle toplanarak Net girdi hesaplanır.
Çarpım $\text{Net} = \prod_{i=1}^{N} X_i * W_i$	Ağırlık değerleri girdiler ile çarpılır ve daha sonra bulunan değerler birbirleriyle çarpılarak Net Girdi Hesaplanır.
Maksimum $\text{Net} = \text{Max}(X_i * W_i)$	n adet girdi içinden ağırlıklar girdilerle çarpıldıktan sonra içlerinden en büyüğü Net girdi olarak kabul edilir.
Minimum $\text{Net} = \text{Min}(X_i * W_i)$	n adet girdi içinden ağırlıklar girdilerle çarpıldıktan sonra içlerinden en küçüğü Net girdi olarak kabul edilir.
Çoğunluk $\text{Net} = \sum_{i=1}^{N} Sgn(X_i * W_i)$	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 $\text{Net} = \text{Net(eski)} + \sum_{i=1}^{N} X_i * W_i$	Hücreye gelen bilgiler ağırlıklı olarak toplanır. Daha önce hücreye gelen bilgilere yeni hesaplanan girdi değerleri eklenerek hücrenin net girdisi hesaplanır.

Aktivasyon Fonksiyonu: Aktivasyon fonksiyonu, toplanan ağırlıklı girdinin nöronun çıkışına eşlenmesidir. Bir aktivasyon/transfer fonksiyonu olarak adlandırılır, çünkü nöronun aktive edildiği ve çıkış sinyalinin gücünün başlangıcını yönetir. Matematiksel olarak, en çok kullanılanları ReLu, tanh, Softmax gibi birçok aktivasyon fonksiyonumuz vardır.

Çıktı Katmanı: Çıktı katmanı tahmin edilen özelliktir, temel olarak oluşturduğunuz modelin türüne bağlıdır.

İleri Beslemeli Derin Ağlar

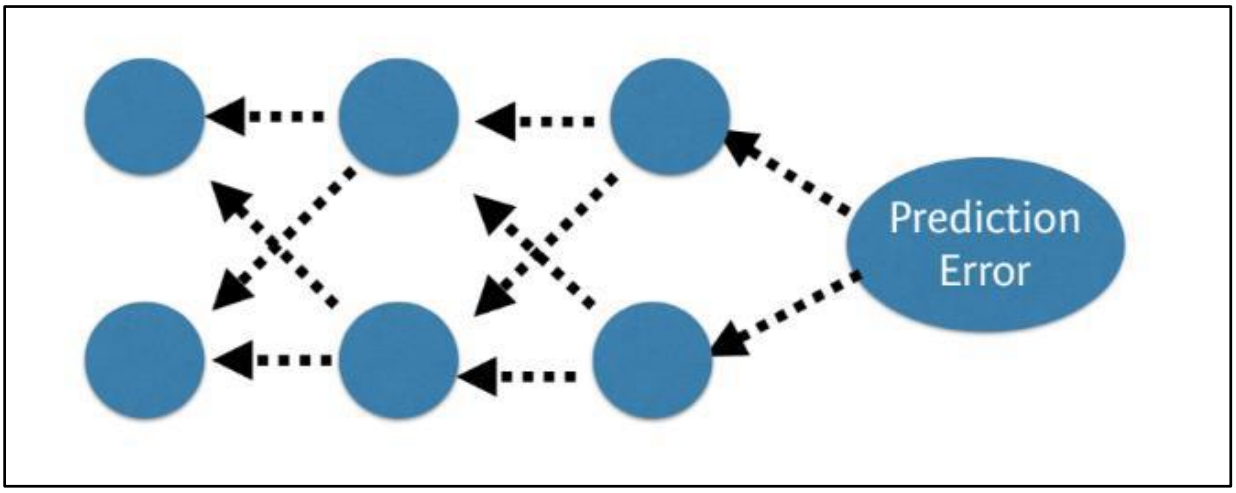
İleri beslemeli denetimli sinir ağları, ilk ve en başarılı öğrenme algoritmaları arasındaydı. Bunlar aynı zamanda derin ağlar, Çok Katmanlı Perceptron (MLP) veya basitçe sinir ağları olarak adlandırılır ve tek bir gizli katmana sahip vanilya mimarisi resmedilir. Her bir nöron, bazı ağırlıklarla diğer nöronla ilişkilidir. Ağ, bir çıkış değeri üretecek şekilde yukarı doğru harekete geçiren nöronları işler. Bu, ağ üzerinde bir ileri geçiş olarak adlandırılır. Şekil 4'te ileri beslemeli yapay sinir ağına örnek verilmiştir.



Şekil 4. İleri beslemeli yapay sinir ağı modeli

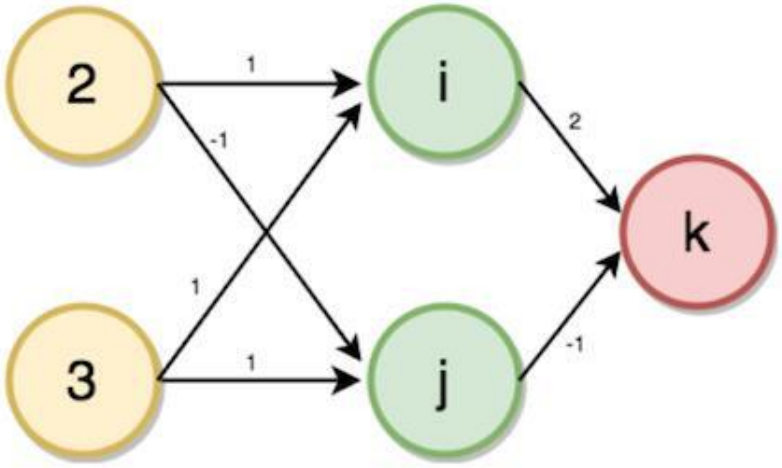
Geri Yayılma Yapay Sinir Ağı

Ağın tahmini değeri beklenen çıktıyla karşılaştırılır ve bir fonksiyon kullanılarak bir hata hesaplanır. Bu hata daha sonra tüm ağda, bir kerede bir katman halinde geri yayılır ve ağırlıklar, hataya katkıda bulundukları değere göre güncellenir. Bu akıllı matematik bitine geri yayılım (back-propagation) algoritması denir. İşlem, eğitim verilerindeki tüm örnekler için tekrarlanır. Şekil 5'te örnek bir geriye yayılım YSA verilmiştir.



Şekil 5. Geriye yayılım yapay sinir ağı modeli.

Örnek uygulama



i = 2 * 1 + 3 * 1 = 5

j = 2 * -1 + 3 * 1 = 1

k = 5 * 2 + 1 * -1 = 9

Örnek Uygulama (YSA ile cep telefonu fiyatlarının artışının belirlenmesi)

Öncelikle gerekli kütüphaneler yüklenir

```
#kütüphanelerin yüklenmesi
import numpy as np
import pandas as pd
from sklearn.preprocessing import LabelEncoder
```

Ardından veriseti yüklenir ve verideki sınıflar etiketlenir.

```
#verisetinin yüklenmesi
veri = pd.read_csv("C:/Users/burak/Masaüstü/YMT410 Yapay Zeka ve Uzman Sistemler/Uygulama #4/telefon_fiy

#Sınıf sayısının belirlenmesi
label_encoder = LabelEncoder().fit(veri.price_range)
labels = label_encoder.transform(veri.price_range)
classes = list(label_encoder.classes_)
```

Girdi ve çıktı verileri hazırlanarak standartlaştırma işlemi yapılır.

```
#girdi ve çıktı verilerinin hazırlanması
X = veri.drop(["price_range"],axis=1)
y = labels

#verilerin standartlaştırılması
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
```

Eğitim ve test verileri hazırlanır ve ardından çıktı değerleri kategorileştirilir.

```
#eğitim ve test verilerinin hazırlanması
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.2)

#çıktı değerlerinin kategorileştirilmesi
from tensorflow.keras.utils import to_categorical
y_train = to_categorical(y_train)
y_test = to_categorical(y_test)
```

Verilerle alakalı olan işlemler tamamlandıktan sonra model oluşturulur.

```
#modelin oluşturulması
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
model = Sequential()
model.add(Dense(16,input_dim=20, activation="relu"))
model.add(Dense(12, activation="relu"))
model.add(Dense(4,activation="softmax"))
model.summary()
```

Son olarak model derlenir ve eğitim işlemine başlanır.

```
#modelin derlenmesi
model. compile(loss="categorical_crossentropy",optimizer="adam",metrics=["accuracy"])

#modelin eğitilmesi
model.fit(X_train,y_train, validation_data=(X_test, y_test), epochs=150)
```

Değerlendirme sonuçları grafik üzerinde gösterilir.

```
#Eğitim ve doğrulama başarımlarının gösterilmesi
import matplotlib.pyplot as plt

plt.plot(model.history.history["accuracy"])
plt.plot(model.history.history["val_accuracy"])
plt.title("Model Başarımları")
plt.ylabel("Başarım")
plt.xlabel("Epok sayısı")
plt.legend(["Eğitim","Test"], loc="upper left")
plt.show()

plt.plot(model.history.history["loss"])
plt.plot(model.history.history["val_loss"])
plt.title("Model Kayıpları")
plt.ylabel("Kayıp")
plt.xlabel("Epok sayısı")
plt.legend(["Eğitim","Test"], loc="upper left")
plt.show()
```

Uygulama 4. Değerlendirme Soruları

1) Verilen kodu inceleyerek kendi modelinizi oluşturunuz **(10p).**
2) Verilen kodu inceleyerek çapraz doğrulama işlemi yapınız ve buna göre başarımı değerlendiriniz **(30p).**
3) Verisetinde (telefon_fiyat_değişimi) toplam 20 adet özellik bulunmaktadır. Bu verisetinden "blue", "fc", "int_memory", "ram" ve "wifi" değerlerini çıkarıp, sınıflandırma işlemini tekrar yapınız **(10p)**.
4) Diyabet verisetini kullanarak bir YSA modeli oluşturunuz. Bu YSA modeline, eğitim ve doğrulama işlemlerinin başarım ve kayıplarını belirleyiniz. Bu değerleri bir grafik üzerinde gösteriniz **(20p).**
5) Soru 4'te oluşturduğunuz modelin ROC eğrisini çiziniz **(30p).**