# Çanakkale Onsekiz Mart Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Akademik Dönem 2022-2023
# Ders: BLM-4014 Yapay Sinir Ağları/Bahar Dönemi
# ARA SINAV SORU VE CEVAP KAĞIDI
# Dersi Veren Öğretim Elemanı: Dr. Öğretim Üyesi Ulya Bayram

Öğrenci Adı Soyadı: Beratcan Güreş
Öğrenci No: 190401093

14 Nisan 2023

## Açıklamalar:

- Vizeyi çözüp, üzerinde aynı sorular, sizin cevaplar ve sonuçlar olan versiyonunu bu formatta PDF olarak, Teams üzerinden açtığım assignment kısmına yüklemeniz gerekiyor. Bu bahsi geçen PDF'i oluşturmak için LaTeX kullandıysanız, tex dosyasının da yer aldığı Github linkini de ödevin en başına (aşağı url olarak) eklerseniz bonus 5 Puan! (Tavsiye: Overleaf)
- Çözümlerde ya da çözümlerin kontrolünü yapmada internetten faydalanmak, ChatGPT gibi servisleri kullanmak serbest. Fakat, herkesin çözümü kendi emeğinden oluşmak zorunda. Çözümlerinizi, cevaplarınızı aşağıda belirttiğim tarih ve saate kadar kimseyle paylaşmayınız.
- Kopyayı önlemek için Github repository'lerinizin hiçbirini **14 Nisan 2023, saat 15:00'a kadar halka açık (public) yapmayınız!** (Assignment son yükleme saati 13:00 ama internet bağlantısı sorunları olabilir diye en fazla ekstra 2 saat daha vaktiniz var. **Fakat 13:00 - 15:00 arası yüklemelerden -5 puan!**
- Ek puan almak için sağlayacağınız tüm Github repository'lerini **en geç 15 Nisan 2023 15:00'da halka açık (public) yapmış olun linklerden puan alabilmek için!**
- **14 Nisan 2023, saat 15:00'dan sonra gönderilen vizeler değerlendirilmeye alınmayacak, vize notu olarak 0 (sıfır) verilecektir!** Son anda internet bağlantısı gibi sebeplerden sıfır almayı önlemek için assignment kısmından ara ara çözümlerinizi yükleyebilirsiniz yedekleme için. Verilen son tarih/saatte (14 Nisan 2023, saat 15:00) sistemdeki en son yüklü PDF geçerli olacak.
- Çözümlerin ve kodların size ait ve özgün olup olmadığını kontrol eden bir algoritma kullanılacaktır. Kopya çektiği belirlenen vizeler otomatikman 0 (sıfır) alacaktır. Bu nedenle çözümlerinizi ve kodlarınızı yukarıda sağladığım gün ve saatlere kadar kimseyle paylaşmayınız.
- Bu vizeden alınabilecek en yüksek not 100'dür. Toplam aldığınız puan 100'ü geçerse, aldığınız not 100'e sabitlenecektir.
- LaTeX kullanarak PDF oluşturanlar öz geçmişlerine LaTeX bildiklerini de eklemeyi unutmasınlar :)
- Bu vizedeki soruların çözümleri ve tex dosyası için istediğiniz kadar sayıda Github repository'si oluşturabilirsiniz. Sadece yukarıda belirttiğim tarihlerde (14 Nisan 2023 saat 15:00 ile 15 Nisan 2023 saat 15:00 arasında) public yapmayı/halka açmayı ve Github profilinizi de öz geçmişinize eklemeyi unutmayın :)
- Bilgisayarınıza hiçbir program kurmadan, Overleaf, Google Colab, Kaggle gibi ortamlar üzerinden bu vizeyi tamamlamanız mümkün. İyi çalışmalar!!

(Ekstra 5 puan) https://github.com/guresberat/ArtificialNeuralNetworksMidTerm

# 1 (Toplam 10 Puan) Hopfield Networks:

**1.1 (2 Puan) ChatGPT'ye size Hopfield Network'ler ile ilgili detaylı bilgi vermesini isteyin. Sorduğunuz soruyu, ChatGPT'nin çıkartığı cevabı aşağı koyun. Bu cevapta anlamadığınız terimleri de ChatGPT'ye sorup, sorduğunuz soruları ve ChatGPT'nin verdiği cevapları da sırayla aşağı koyun.**

Hopfield Network'ler, adını John Hopfield'dan almıştır ve ilk olarak 1982 yılında tanıtılmıştır. Ağ, bir dizi nöron ve her nöronun birbirine bağlı olduğu bir matris ile tanımlanır. Bu matris, ağırlık matrisi olarak adlandırılır ve nöronların birbirleriyle etkileşimini belirler.

Ağın işleyişi, basitçe şöyle özetlenebilir: Öncelikle, ağa bir dizi girdi verilir. Bu girdiler, ağırlık matrisindeki ilgili nöronlarla çarpılır ve sonuçlar toplanarak bir çıkış değeri üretilir. Bu çıkış değeri, tekrar ağın girdisi olarak kullanılır ve bu işlem belirli bir çıkışa kadar tekrarlanır.

Hopfield Network'lerin temel özelliklerinden biri, hatırlama özelliğidir. Ağ, bir dizi girdi örneği öğrenir ve bu örnekleri saklar. Daha sonra ağa benzer girdiler verildiğinde, ağ öğrendiği örneklerden en yakın olanını hatırlar ve uygun bir çıkış üretir.

Ayrıca, Hopfield Network'ler aynı zamanda enerji fonksiyonlarına da sahiptirler. Bu enerji fonksiyonları, ağın girdilerinin belirli bir şekilde düzenlenmesiyle azaltılabilir. Bu özellik, ağırlık matrisinin hesaplanmasında da kullanılır.

Hopfield Network'ler, çeşitli uygulamalarda kullanılmaktadır. Örneğin, resim tanıma ve yüz tanıma gibi görsel tanıma uygulamalarında kullanılabilirler. Ayrıca, özellikle hafıza özellikleri nedeniyle, hastalıkların teşhisinde ve hatta nöroprotezlerin geliştirilmesinde de kullanılabilmektedirler.

**1.2 (8 Puan) ChatGPT'nin açıklamalarından ne anladığınızı buraya kendi kelimelerinizle özetleyin. Örneğin "Hopfield Network nedir? Ne işe yarar? Neden bilmemiz gerekir bunu? Gerçek hayatta kullanılıyor mu?" gibi. Anlamadığınız kısımlar varsa ve ChatGPT'nin açıklamaları yeterli gelmezse internet dahil farklı kaynaklardan araştırıp, bilgilerin doğruluğunu sorgulamakta serbestsiniz. Konuyu doğru anlayıp anlamamanız puana etki edecektir. Kullandığınız kaynakları da belirtin!**

Hopfield ağları, yapay sinir ağları ailesine ait bir türdür ve özellikle bellek işlemleri ve optimizasyon problemleri gibi alanlarda kullanılır. Bu ağlar, birçok nöronun etkileşim halinde olduğu bir özyineleme yapısı kullanarak verileri depolayabilir ve hatırlayabilir. Hopfield ağları, çeşitli uygulama alanlarında kullanılır ve özellikle yapay zeka ve robotik gibi alanlarda önemlidir.

Kaynak: https://en.wikipedia.org/wiki/Hopfield$_n etwork$

# 2 (Toplam 10 Puan) Boltzman Machine:

**2.1 (2 Puan) ChatGPT'ye size Boltzman Machine ilgili detaylı bilgi vermesini isteyin. Sorduğunuz soruyu, ChatGPT'nin çıkartığı cevabı aşağı koyun. Bu cevapta anlamadığınız terimleri de ChatGPT'ye sorup, sorduğunuz soruları ve ChatGPT'nin verdiği cevapları da sırayla aşağı koyun.**

Boltzmann Machine, yapay sinir ağı modellerinden biridir. Çok sayıda girdi ve çıktı katmanı arasında gizli katmanlardan oluşur. Gizli katmanlardaki birimler, birbirleriyle bağlantılıdır ve bunların ağırlıkları, ağın öğrenmesi sırasında belirlenir.

Boltzmann Machine, enerji tabanlı bir modeldir. Ağın öğrenmesi, enerji fonksiyonunun minimize edilmesi esasına dayanır. Ağın belirli bir durumda enerjisi, o durumun ne kadar doğru veya yanlış olduğunu belirler.

Bir Boltzmann Machine, iki tür birime sahiptir: gizli birimler ve görünen birimler. Gizli birimler, çıktı katmanı ve girdi katmanı arasında yer alan katmanlardır ve görünen birimler, ağın girdileridir. Bir

Boltzmann Machine, her bir gizli birim için bir eşik değerine ve her bir görünen birim için bir ağırlık değerine sahiptir. Bu değerler, ağın öğrenmesi sırasında belirlenir.

Bir Boltzmann Machine, iki temel durumda çalışır: öğrenme ve tanıma. Öğrenme, ağırlıkların ve eşik değerlerinin belirlenmesi sürecidir. Tanıma, ağın belirli bir durumda, verilen girdiye en uygun çıktıyı üretmesi sürecidir.

**2.2 (8 Puan) ChatGPT'nin açıklamalarından ne anladığınızı buraya kendi kelimelerinizle özetleyin. Örneğin "Boltzman Machine nedir? Ne işe yarar? Neden bilmemiz gerekir bunu? Gerçek hayatta kullanılıyor mu?" gibi. Anlamadığınız kısımlar varsa ve ChatGPT'nin açıklamaları yeterli gelmezse internet dahil farklı kaynaklardan araştırıp, bilgilerin doğruluğunu sorgulamakta serbestsiniz. Konuyu doğru anlayıp anlamamanız puana etki edecektir. Kullandığınız kaynakları da belirtin!**

Boltzmann Makinesi, yapay sinir ağları ailesine ait bir türdür ve özellikle olasılıksal modellerin öğrenilmesi için kullanılır. Bu ağlar, birçok nöronun etkileşim halinde olduğu bir özyineleme yapısı kullanarak verileri analiz edebilir ve modelleyebilir. Boltzmann Makineleri, unsupervised learning (gözetimsiz öğrenme) için kullanılır ve özellikle veri boyutunu azaltma, özellik çıkarımı ve örüntü tanıma gibi alanlarda kullanılır. Boltzmann Makinesi, Hinton ve Sejnowski tarafından 1985 yılında tanıtılmıştır ve o zamandan beri birçok farklı uygulama alanında kullanılmıştır.

Kaynak : https://en.wikipedia.org/wiki/Boltzmann$_machine$

# 3 (Toplam 10 Puan) Markov Assumption ve Markov Chain:

**3.1 (2 Puan) ChatGPT'ye size Markov assumption ve Markov Chain ilgili detaylı bilgi vermesini isteyin. Sorduğunuz soruyu, ChatGPT'nin çıkartığı cevabı aşağı koyun. Bu cevapta anlamadığınız terimleri de ChatGPT'ye sorup, sorduğunuz soruları ve ChatGPT'nin verdiği cevapları da sırayla aşağı koyun.**

Markov assumption, bir sistemin gelecekteki durumunun, sadece mevcut durumuna bağlı olduğunu ve geçmiş durumlarla ilgisi olmadığını varsayar. Başka bir deyişle, bir olayın olasılığı, sadece son gözlemlenen olaya bağlıdır ve önceki olaylar göz önünde bulundurulmaz. Bu varsayım, Markov Chain modellerinin temelini oluşturur.

Markov Chain, bir dizi durumun, Markov assumption'a göre hareket ettiği olasılıksal bir modeldir. Markov Chain'in temel özelliği, sadece geçerli durumun, gelecekteki durumların olasılığı üzerinde etkili olmasıdır. Geçmiş durumlar, gelecekteki durumların belirlenmesinde önemli değildir. Bu nedenle Markov Chain, sadece son durumu kullanarak gelecekteki olasılıkları tahmin edebilir.

Markov Chain, olasılık teorisi ve istatistikte kullanılan bir araçtır ve çeşitli uygulamalarda kullanılır. Örneğin, finansal piyasalardaki fiyat hareketlerinin tahmininde, doğal dil işlemede, zaman serilerindeki modellerde, moleküler dinamiklerde ve birçok diğer alanda Markov Chain modelleri kullanılabilir.

Markov Chain'in temel matematiksel yapısı, geçiş olasılık matrisi (transition probability matrix) ve denge durumu (steady state) kavramlarını içerir. Geçiş olasılık matrisi, bir durumdan diğerine geçiş olasılıklarını içeren bir matristir. Denge durumu ise, zincirin zaman içinde belirli bir duruma ulaştığında durumun sabit kalacağı koşulları ifade eder.

**3.2 (8 Puan) ChatGPT'nin açıklamalarından ne anladığınızı buraya kendi kelimelerinizle özetleyin. Örneğin "Markov assumption ve Markov Chain nedir? Ne işe yarar? Neden bilmemiz gerekir bunu? Gerçek hayatta kullanılıyor mu?" gibi. Anlamadığınız kısımlar varsa ve ChatGPT'nin açıklamaları yeterli gelmezse internet dahil farklı kaynaklardan araştırıp, bilgilerin doğruluğunu sorgulamakta serbestsiniz. Konuyu doğru anlayıp anlamamanız puana etki edecektir. Kullandığınız kaynakları da belirtin!**

Markov varsayımı, bir sistemin gelecekteki durumunun sadece mevcut durumdan ve mevcut durumda alınan aksiyonlardan bağımsız olduğunu varsayar. Bu varsayım, birçok alanda, özellikle makine öğrenmesi, yapay zeka ve kontrol teorisi gibi alanlarda yaygın bir şekilde kullanılır. Markov varsayımı sayesinde, sistemin gelecekteki durumunu tahmin etmek ve kontrol etmek daha kolay hale gelir.

Kaynak : https://en.wikipedia.org/wiki/Markov$_p roperty$

# 4 (Toplam 20 Puan) Feed Forward:

- Forward propagation için, input olarak şu X matrisini verin (tensöre çevirmeyi unutmayın): $X = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ Satırlar veriler (sample'lar), kolonlar öznitelikler (feature'lar).
- Bir adet hidden layer olsun ve içinde tanh aktivasyon fonksiyonu olsun
- Hidden layer'da 50 nöron olsun
- Bir adet output layer olsun, tek nöronu olsun ve içinde sigmoid aktivasyon fonksiyonu olsun

Tanh fonksiyonu:
$f(x) = \frac{exp(x)-exp(-x)}{exp(x)+exp(-x)}$

Sigmoid fonksiyonu:
$f(x) = \frac{1}{1+exp(-x)}$

**Pytorch kütüphanesi ile, ama kütüphanenin hazır aktivasyon fonksiyonlarını kullanmadan, formülünü verdiğim iki aktivasyon fonksiyonunun kodunu ikinci haftada yaptığımız gibi kendiniz yazarak bu yapay sinir ağını oluşturun ve aşağıdaki üç soruya cevap verin.**

## 4.1 (10 Puan) Yukarıdaki yapay sinir ağını çalıştırmadan önce pytorch için Seed değerini 1 olarak set edin, kodu aşağıdaki kod bloğuna ve altına da sonucu yapıştırın:

```
import torch
import torch.nn as nn

X = torch.tensor([[1, 2, 3], [4, 5, 6]], dtype=torch.float)
torch.manual_seed(1)

def tanh(x):
  return (torch.exp(x) - torch.exp(-x)) / (torch.exp(x) + torch.exp(-x))

def sigmoid(x):
  return 1 / (1 + torch.exp(-x))

class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.hidden = nn.Linear(3, 50)
        self.output = nn.Linear(50, 1)

    def forward(self, x):
        x = tanh(self.hidden(x))
        x = sigmoid(self.output(x))
        return x

model = Model()
output = model(X)
print(output)
```

tensor([[0.4892],[0.5566]])

**4.2 (5 Puan) Yukarıdaki yapay sinir ağını çalıştırmadan önce Seed değerini öğrenci numaranız olarak değiştirip, kodu aşağıdaki kod bloğuna ve altına da sonucu yapıştırın:**

```
import torch
import torch.nn as nn

X = torch.tensor([[1, 2, 3], [4, 5, 6]], dtype=torch.float)
torch.manual_seed(190401093)

def tanh(x):
  return (torch.exp(x) - torch.exp(-x)) / (torch.exp(x) + torch.exp(-x))

def sigmoid(x):
  return 1 / (1 + torch.exp(-x))

class Model(nn.Module):
    def __init__(self):
        super(Model, self).__init__()
        self.hidden = nn.Linear(3, 50)
        self.output = nn.Linear(50, 1)

    def forward(self, x):
        x = tanh(self.hidden(x))
        x = sigmoid(self.output(x))
        return x

model = Model()
output = model(X)
print(output)
```

tensor([[0.5900],[0.5951]])

**4.3 (5 Puan) Kodlarınızın ve sonuçlarınızın olduğu jupyter notebook'un Github repository'sindeki linkini aşağıdaki url kısmının içine yapıştırın. İlk sayfada belirttiğim gün ve saate kadar halka açık (public) olmasın:**

https://github.com/guresberat/ArtificialNeuralNetworksMidTerm/blob/master/YapaySinirAg%CC%86lar%C4%B1Soru4.ipynb

# 5 (Toplam 40 Puan) Multilayer Perceptron (MLP):

**Bu bölümdeki sorularda benim vize ile beraber paylaştığım Prensesi İyileştir (Cure The Princess) Veri Seti parçaları kullanılacak. Hikaye şöyle (soruyu çözmek için hikaye kısmını okumak zorunda değilsiniz):**

"Bir zamanlar, çok uzaklarda bir ülkede, ağır bir hastalığa yakalanmış bir prenses yaşarmış. Ülkenin kralı ve kraliçesi onu iyileştirmek için ellerinden gelen her şeyi yapmışlar, ancak denedikleri hiçbir çare işe yaramamış.

Yerel bir grup köylü, herhangi bir hastalığı iyileştirmek için gücü olduğu söylenen bir dizi sihirli malzemeden bahsederek kral ve kraliçeye yaklaşmış. Ancak, köylüler kral ile kraliçeyi, bu malzemelerin etkilerinin patlayıcı olabileceği ve son zamanlarda yaşanan kuraklıklar nedeniyle bu malzemelerden sadece birkaçının herhangi bir zamanda bulunabileceği konusunda uyarmışlar. Ayrıca, sadece deneyimli bir simyacı bu özelliklere sahip patlayıcı ve az bulunan malzemelerin belirli bir kombinasyonunun prensesi iyileştireceğini belirleyebilecekmiş.

Kral ve kraliçe kızlarını kurtarmak için umutsuzlar, bu yüzden ülkedeki en iyi simyacıyı bulmak için yola çıkmışlar. Dağları tepeleri aşmışlar ve nihayet "Yapay Sinir Ağları Uzmanı" olarak bilinen yeni bir sihirli sanatın ustası olarak ün yapmış bir simyacı bulmuşlar.

Simyacı önce köylülerin iddialarını ve her bir malzemenin alınan miktarlarını, ayrıca iyileşmeye yol açıp açmadığını incelemiş. Simyacı biliyormuş ki bu prensesi iyileştirmek için tek bir şansı varmış ve bunu doğru yapmak zorundaymış. (Original source: https://www.kaggle.com/datasets/unmoved/cure-the-princess)

(Buradan itibaren ChatGPT ve Dr. Ulya Bayram'a ait hikayenin devamı)

Simyacı, büyülü bileşenlerin farklı kombinasyonlarını analiz etmek ve denemek için günler harcamış. Sonunda birkaç denemenin ardından prensesi iyileştirecek çeşitli karışım kombinasyonları bulmuş ve bunları bir veri setinde toplamış. Daha sonra bu veri setini eğitim, validasyon ve test setleri olarak üç parçaya ayırmış ve bunun üzerinde bir yapay sinir ağı eğiterek kendi yöntemi ile prensesi iyileştirme ihtimalini hesaplamış ve ikna olunca kral ve kraliçeye haber vermiş. Heyecanlı ve umutlu olan kral ve kraliçe, simyacının prensese hazırladığı ilacı vermesine izin vermiş ve ilaç işe yaramış ve prenses hastalığından kurtulmuş.

Kral ve kraliçe, kızlarının hayatını kurtardığı için simyacıya krallıkta kalması ve çalışmalarına devam etmesi için büyük bir araştırma bütçesi ve çok sayıda GPU'su olan bir server vermiş. İyileşen prenses de kendisini iyileştiren yöntemleri öğrenmeye merak salıp, krallıktaki üniversitenin bilgisayar mühendisliği bölümüne girmiş ve mezun olur olmaz da simyacının yanında, onun araştırma grubunda çalışmaya başlamış. Uzun yıllar birlikte krallıktaki insanlara, hayvanlara ve doğaya faydalı olacak yazılımlar geliştirmişler, ve simyacı emekli olduğunda prenses hem araştırma grubunun hem de krallığın lideri olarak hayatına devam etmiş.

Prenses, kendisini iyileştiren veri setini de, gelecekte onların izinden gidecek bilgisayar mühendisi prensler ve prensesler başkalarına faydalı olabilecek yapay sinir ağları oluşturmayı öğrensinler diye halka açmış ve sınavlarda kullanılmasını salık vermiş."

**İki hidden layer'lı bir Multilayer Perceptron (MLP) oluşturun beşinci ve altıncı haftalarda yaptığımız gibi. Hazır aktivasyon fonksiyonlarını kullanmak serbest. İlk hidden layer'da 100, ikinci hidden layer'da 50 nöron olsun. Hidden layer'larda ReLU, output layer'da sigmoid aktivasyonu olsun.**

**Output layer'da kaç nöron olacağını veri setinden bakıp bulacaksınız. Elbette bu veriye uygun Cross Entropy loss yöntemini uygulayacaksınız. Optimizasyon için Stochastic Gradient Descent yeterli. Epoch sayınızı ve learning rate'i validasyon seti üzerinde denemeler yaparak (loss'lara overfit var mı diye bakarak) kendiniz belirleyeceksiniz. Batch size'ı 16 seçebilirsiniz.**

### 5.1 (10 Puan) Bu MLP'nin pytorch ile yazılmış class'ının kodunu aşağı kod bloğuna yapıştırın:

```
class MLP(nn.Module):
    def __init__(self, input_size, hidden_size1, hidden_size2, output_size):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size1)
        self.relu1 = nn.ReLU()
        self.fc2 = nn.Linear(hidden_size1, hidden_size2)
        self.relu2 = nn.ReLU()
        self.fc3 = nn.Linear(hidden_size2, output_size)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        out = self.fc1(x)
        out = self.relu1(out)
        out = self.fc2(out)
        out = self.relu2(out)
        out = self.fc3(out)
        out = self.sigmoid(out)
        return out
```

**5.2 (10 Puan) SEED=öğrenci numaranız set ettikten sonra altıncı haftada yazdığımız gibi training batch'lerinden eğitim loss'ları, validation batch'lerinden validasyon loss değerlerini hesaplayan kodu aşağıdaki kod bloğuna yapıştırın ve çıkan figürü de alta ekleyin.**

```
import numpy as np
import pandas as pd
import torch
import torchvision
import torchvision.transforms as transforms
import torch.optim as optim
import torch.nn as nn
import seaborn as sns
import torch.nn.functional as F
import matplotlib.pyplot as plt

# Load the data
train_data = pd.read_csv('/content/cure_the_princess_train.csv')
val_data = pd.read_csv('/content/cure_the_princess_validation.csv')
test_data = pd.read_csv('/content/cure_the_princess_test.csv')

# Convert the data to tensors
x_train = torch.tensor(train_data.iloc[:, :-1].values, dtype=torch.float)
y_train = torch.tensor(train_data.iloc[:, -1].values, dtype=torch.float).unsqueeze(1)
x_val = torch.tensor(val_data.iloc[:, :-1].values, dtype=torch.float)
y_val = torch.tensor(val_data.iloc[:, -1].values, dtype=torch.float).unsqueeze(1)
x_test = torch.tensor(test_data.iloc[:, :-1].values, dtype=torch.float)
y_test = torch.tensor(test_data.iloc[:, -1].values, dtype=torch.float).unsqueeze(1)

# Hyperparameters
epochs = 10
batch_size = 16
learning_rate = 0.1

# MLP olu turma
class MLP(nn.Module):
    def __init__(self, input_size, hidden_size1, hidden_size2, output_size):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size1)
        self.relu1 = nn.ReLU()
        self.fc2 = nn.Linear(hidden_size1, hidden_size2)
        self.relu2 = nn.ReLU()
        self.fc3 = nn.Linear(hidden_size2, output_size)
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        out = self.fc1(x)
        out = self.relu1(out)
        out = self.fc2(out)
        out = self.relu2(out)
        out = self.fc3(out)
        out = self.sigmoid(out)
        return out

input_size = x_train.shape[1]
hidden_size1 = 100
hidden_size2 = 50
output_size = 1
torch.manual_seed(190401093)
mlp = MLP(input_size, hidden_size1, hidden_size2, output_size)

# Optimizer ve Loss fonksiyonlar  tan mlama
optimizer = optim.SGD(mlp.parameters(), lr=learning_rate)
criterion = nn.BCEWithLogitsLoss()

num_epochs = 1000
```

```
train_losses = []
val_losses = []
test_losses = []
train_accs = []
val_accs = []
test_accs = []
for epoch in range(num_epochs):
    # Training mode
    mlp.train()

    # Forward pass
    y_pred_train = mlp(x_train)
    train_loss = criterion(y_pred_train, y_train)
    train_losses.append(train_loss.item())

    # Backward pass and parameter update
    optimizer.zero_grad()
    train_loss.backward()
    optimizer.step()

    # Evaluation mode
    mlp.eval()

    # Forward pass on validation data
    with torch.no_grad():
        y_pred_val = mlp(x_val)
        val_loss = criterion(y_pred_val, y_val)
        val_losses.append(val_loss.item())

    with torch.no_grad():
        y_pred_test = mlp(x_test)
        test_loss = criterion(y_pred_test, y_test)
        test_losses.append(test_loss.item())
```

**5.3 (10 Puan) SEED=öğrenci numaranız set ettikten sonra altıncı haftada ödev olarak verdiğim gibi earlystopping'deki en iyi modeli kullanarak, Prensesi İyileştir test setinden accuracy, F1, precision ve recall değerlerini hesaplayan kodu yazın ve sonucu da aşağı yapıştırın. %80'den fazla başarı bekliyorum test setinden. Daha düşükse başarı oranınız, nerede hata yaptığınızı bulmaya çalışın. %90'dan fazla başarı almak mümkün (ben denedim).**

```
def evaluate(model, x, y):
    with torch.no_grad():
        y_pred = model(x)
        y_pred = (y_pred > 0.5).float()
        total_acc = (y_pred == y).float().mean().item()

        tp = ((y_pred == 1) & (y == 1)).float().sum().item()
        fp = ((y_pred == 1) & (y == 0)).float().sum().item()
        tn = ((y_pred == 0) & (y == 0)).float().sum().item()
        fn = ((y_pred == 0) & (y == 1)).float().sum().item()

        recall = tp / (tp + fn)
        precision = tp / (tp + fp)
        f1 = 2 * (precision * recall) / (precision + recall)

        loss = F.binary_cross_entropy(y_pred, y)

        return total_acc, recall, precision, f1, loss

train_acc, train_recall, train_precision, train_f1, train_loss = evaluate(mlp,
                                                    x_train, y_train)
val_acc, val_recall, val_precision, val_f1, val_loss = evaluate(mlp, x_val, y_val)
```

```
test_acc, test_recall, test_precision, test_f1, test_loss = evaluate(mlp, x_test,
                                                    y_test)

print(f"Train Accuracy: {train_acc:.4f} Recall: {train_recall:.4f} Precision: {
                                                    train_precision:.4f} F1: {train_f1:.4f}
                                                    Loss: {train_loss:.4f}")
print(f"Validation Accuracy: {val_acc:.4f} Recall: {val_recall:.4f} Precision: {
                                                    val_precision:.4f} F1: {val_f1:.4f} Loss:
                                                    {val_loss:.4f}")
print(f"Test Accuracy: {test_acc:.4f} Recall: {test_recall:.4f} Precision: {
                                                    test_precision:.4f} F1: {test_f1:.4f} Loss
                                                    : {test_loss:.4f}")
```

Train Accuracy: 0.9529 Recall: 0.9238 Precision: 0.9794 F1: 0.9508 Loss: 4.7125 Validation Accuracy: 0.9299 Recall: 0.8910 Precision: 0.9653 F1: 0.9267 Loss: 7.0064 Test Accuracy: 0.9301 Recall: 0.8711 Precision: 0.9883 F1: 0.9260 Loss: 6.9948

**5.4 (5 Puan) Tüm kodların CPU'da çalışması ne kadar sürüyor hesaplayın. Sonra to device yöntemini kullanarak modeli ve verileri GPU'ya atıp kodu bir de böyle çalıştırın ve ne kadar sürdüğünü hesaplayın. Süreleri aşağıdaki tabloya koyun. GPU için Google Colab ya da Kaggle'ı kullanabilirsiniz, iki ortam da her hafta saatlerce GPU hakkı veriyor.**

Tablo 1: Ortalama Çalışma Süreleri

| Ortam | Süre (saniye) |
|---|---|
| CPU | 3.905 |
| GPU | 3.171 |

**5.5 (3 Puan) Modelin eğitim setine overfit etmesi için elinizden geldiği kadar kodu gereken şekilde değiştirin, validasyon loss'unun açıkça yükselmeye başladığı, training ve validation loss'ları içeren figürü aşağı koyun ve overfit için yaptığınız değişiklikleri aşağı yazın. Overfit, tam bir çanak gibi olmalı ve yükselmeli. Ona göre parametrelerle oynayın.**

Cevaplar buraya

**5.6 (2 Puan) Beşinci soruya ait tüm kodların ve cevapların olduğu jupyter notebook'un Github linkini aşağıdaki url'e koyun.**

https://github.com/guresberat/ArtificialNeuralNetworksMidTerm/blob/master/YapaySinirAg%CC%86lar%C4%B1Soru5_6.ipynb

# 6 (Toplam 10 Puan)

**Bir önceki sorudaki Prensesi İyileştir problemindeki yapay sinir ağınıza seçtiğiniz herhangi iki farklı regülarizasyon yöntemi ekleyin ve aşağıdaki soruları cevaplayın.**

**6.1 (2 puan) Kodlarda regülarizasyon eklediğiniz kısımları aşağı koyun:**

```
# Train the model
num_epochs = 1000
train_losses = []
val_losses = []
test_losses = []
train_accs = []
val_accs = []
test_accs = []
for epoch in range(num_epochs):
    # Training mode
```

```
    mlp.train()

    # Forward pass
    y_pred_train = mlp(x_train)
    train_loss = criterion(y_pred_train, y_train)

    # L1 regularization
    l1_lambda = 0.001
    l1_reg = torch.tensor(0.)
    for param in mlp.parameters():
        l1_reg += torch.norm(param, 1)
    train_loss += l1_lambda * l1_reg

    # L2 regularization
    l2_lambda = 0.001
    l2_reg = torch.tensor(0.)
    for param in mlp.parameters():
        l2_reg += torch.norm(param, 2)
    train_loss += l2_lambda * l2_reg

    train_losses.append(train_loss.item())

    # Backward pass and parameter update
    optimizer.zero_grad()
    train_loss.backward()
    optimizer.step()

    # Evaluation mode
    mlp.eval()

    # Forward pass on validation data
    with torch.no_grad():
        y_pred_val = mlp(x_val)
        val_loss = criterion(y_pred_val, y_val)

        # L1 regularization
        l1_reg = torch.tensor(0.)
        for param in mlp.parameters():
            l1_reg += torch.norm(param, 1)
        val_loss += l1_lambda * l1_reg

        # L2 regularization
        l2_reg = torch.tensor(0.)
        for param in mlp.parameters():
            l2_reg += torch.norm(param, 2)
        val_loss += l2_lambda * l2_reg

        val_losses.append(val_loss.item())
```

### 6.2 (2 puan) Test setinden yeni accuracy, F1, precision ve recall değerlerini hesaplayıp aşağı koyun:

Train Accuracy: 0.9537 Recall: 0.9271 Precision: 0.9778 F1: 0.9517 Loss: 4.6326 Validation Accuracy: 0.9490 Recall: 0.9295 Precision: 0.9667 F1: 0.9477 Loss: 5.0955 Test Accuracy: 0.9430 Recall: 0.9124 Precision: 0.9725 F1: 0.9415 Loss: 5.6995

### 6.3 (5 puan) Regülarizasyon yöntemi seçimlerinizin sebeplerini ve sonuçlara etkisini yorumlayın:

L1 Loss ve MSE Loss, regülarizasyon olarak kullanıldıklarında ağırlıkların büyüklüğünü sınırlarlar ve aşırı uyum (overfitting) problemini önlerler.

L1 Loss, L1 norm regülarizasyonu olarak da bilinir ve ağırlıkların mutlak değerlerinin toplamını mini-

mize ederek çalışır. Bu nedenle, ağırlıkların bazılarını sıfıra yakınlaştırarak gereksiz özelliklerin filtrelenmesine ve modelin daha az karmaşık hale gelmesine yardımcı olur.

MSE Loss, L2 norm regülarizasyonu olarak da bilinir ve ağırlıkların karelerinin toplamını minimize ederek çalışır. Bu nedenle, büyük ağırlıkların cezalandırılmasını sağlar, böylece ağın daha az hassas hale gelmesine ve overfittingin önlenmesine yardımcı olur.

Bu nedenle, bu regülarizasyonlar modelin genelleştirme performansını arttırır ve overfittingi önleyerek daha iyi sonuçlar elde etmenize yardımcı olur.

### 6.4 (1 puan) Sonucun github linkini aşağıya koyun:

https://github.com/guresberat/ArtificialNeuralNetworksMidTerm/blob/master/YapaySinirAg%CC%86lar%C4%B1Soru5_6.ipynb