

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: Ahmet Salih AYMELEK
Öğrenci No: 190401068

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 sağlayacağımız 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 aldığımız puan 100'ü geçerse, aldığımız 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/ahmetsalihaymelek/Yapay_sinir_aglari/blob/main/yapay_sinir_aglari%20\(3\).zip](https://github.com/ahmetsalihaymelek/Yapay_sinir_aglari/blob/main/yapay_sinir_aglari%20(3).zip)

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 çıkarttığı cevabı aşağı koyun. Bu cevapta anlamadığınız terimleri de ChatGPT'ye sorup, sorduğunuz soruları ve ChatGPT'nin verdiği cevapları da sırayla aşağı koyun.

SORU : Hopfield Network'ler ile ilgili detaylı bilgi verirmisin

CEVAP: Hopfield ağları (Hopfield Networks) bir tür geri beslemeli sinir ağıdır. Bu ağlar, özellikle küçük boyutlu problemlerde kullanılan özyinelemeli (recurrent) ağlardan biridir. Hopfield ağları, genellikle modelleme ve eniyileme gibi problemlerin çözümünde kullanılır.

Hopfield ağları, McCulloch-Pitts nöronlarına benzer bir modele dayanır. Ancak, her nöronun çıkışı tüm diğer nöronlara bağlanır ve böylece her bir nöron ağı tümünü temsil eder. Bu özellik, her bir nöronun ağındaki durumu temsil ettiği anlamına gelir. Hopfield ağları, enerji minimizasyonu veya daha spesifik olarak bir enerji fonksiyonunun minimize edilmesi yoluyla problemleri çözer.

Hopfield ağları, iki modda çalışabilir: öğrenme modu ve çalışma modu. Öğrenme modunda, ağa bir dizi örnek veri sunulur ve ağ, bu verileri hatırlamak için ağırlıklarını ayarlar. Çalışma modunda, ağa başlangıç durumu verilir ve ağ, verilen durumdan enerji fonksiyonunu minimize ederek çıkmaya çalışır.

Hopfield ağları, birçok uygulama alanında kullanılabilir. Örneğin, desen tanıma, veri sıkıştırma, veri düzeltme, optimizasyon gibi problemlerin çözümünde kullanılabilir. Ancak, Hopfield ağlarının bellek kapasitesi sınırlıdır ve büyük boyutlu problemlerde uygulanamazlar.

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!

CEVAP: Küçük boyutlu problemlerde kullanılan bir ağıdır. Hopfield ağları, enerji minimizasyonu veya daha spesifik olarak bir enerji fonksiyonunun minimize edilmesi yoluyla problemleri çözer. Hopfield ağları, bellek kapasiteleri sınırlı olduğu için genellikle küçük ölçekli problemlerin çözümünde kullanılırlar. Desen Tanıma, Optimizasyon, Veri Sıkıştırma, Veri Düzeltme, Optik Karakter Tanıma, Biyo-medikal Mühendislik gibi alanlar, Hopfield ağları kullanım alanlarından bazılarıdır Kaynak(Chatgpt)

2 (Toplam 10 Puan) Boltzman Machine:

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

SORU: Boltzman Machine hakkında detaylı bilgi verirmisin?

CEVAP: Boltzmann Makinesi, 1980'lerde tanıtılan bir yapay sinir ağı modelidir. Bu model, gizli ve gözlemlenen değişkenler arasındaki olasılıklı ilişkileri modelleyerek verilerin özelliklerini belirler. Boltzmann Makinesi, en temel haliyle iki tip değişkenden oluşur: gözlemlenen değişkenler ve gizli değişkenler. Gözlemlenen değişkenler, verilerin kendisidir, gizli değişkenler ise bu verilerin özelliklerini temsil eder.

Boltzmann Makinesi'nin en temel işlevi, veriler arasındaki doğrusal olmayan ilişkileri modellemektir. Bu modelleme işlemi, olasılıklı bağlantılar sayesinde gerçekleştirilir. Her bir bağlantı, bir ağırlık ile temsil edilir ve bu ağırlıklar, öğrenme süreci sırasında belirlenir. Boltzmann Makinesi, özellikle veri boyutunu azaltmak ve verilerin özniteliklerini belirlemek için kullanılır.

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!

CEVAP:Boltzmann makineleri, yapay sinir ağlarının bir türüdür ve genellikle gizli katmanlardan oluşur. Bu ağlar, öğrenme sırasında veriler arasındaki ilişkileri öğrenir ve veriler arasındaki olası birlikteliklerin bir modelini oluşturur.

Boltzmann makineleri, sinir ağlarındaki her bir nöronun etkinliği arasındaki bağlantıları ölçmek için bir enerji fonksiyonu kullanır.

Boltzmann makinelerini öğrenmek önemlidir, çünkü bu makineler birçok makine öğrenimi uygulamasında kullanılabilir.Örneğin Gerçek hayatta Boltzmann makineleri, yapay zeka, nörobilim, finansal tahmin ve diğer birçok alanda kullanılabilir.Örneğin, Boltzmann makineleri, beynin işleyişini modeller, kullanıcıların tercihlerini belirleyebilir ve finansal piyasaların davranışını tahmin etmek için kullanılabilir.

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 çıkarttığı cevabı aşağı koyun. Bu cevapta anlamadığınız terimleri de ChatGPT’ye sorup, sorduğunuz soruları ve ChatGPT’nin verdiği cevapları da sırayla aşağı koyun.

Markov zincirleri, bir dizi olayı veya durumu açıklamak için kullanılan istatistiksel bir modeldir. Markov zincirlerinde, her bir olay veya durum, yalnızca önceki olay veya durum tarafından belirlenir ve gelecekteki olaylar veya durumlar, yalnızca mevcut durumun belirli bir özelliği tarafından belirlenir. Bu nedenle, herhangi bir verilen durum veya olay, yalnızca son durum veya olaya dayalı olarak tahmin edilebilir.(Chatgpt)

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 assumption, bir sistemin gelecekteki durumunun sadece mevcut durumuyla ilgilendiği varsayımdır.Bir sistemin gelecekteki durumu, sadece mevcut durumuna bağlıdır ve sistemin geçmişi veya diğer faktörleri bu durumu etkilemez.

Markov Chain,Markov assumption’ı kullanarak modellenen matematiksel bir yapıdır.

Günlük hayatta doğal dil işleme uygulamalarında, kelime sıralarının olasılıklarını modellemek için kullanılırlar.Finansal piyasalardaki fiyat hareketlerini modellemek için de kullanılırlar

4 (Toplam 20 Puan) Feed Forward:

- Forward propagation için, input olarak şu X matrisini verin (tensöre çevirmeyi unutmayın):

$$X = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \text{ Satırlar veriler (sample'lar), kolonlar öznitelikler (feature'lar).}$$

- 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
import torch.optim as optim
torch.manual_seed(1)

X = torch.tensor([[1, 2, 3], [4, 5, 6]], dtype=torch.float32)
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(3, 50) # input boyutu 3, hidden layer boyutu 50
        #self.tanh = nn.Tanh() # tanh aktivasyon fonksiyonu
        self.fc2 = nn.Linear(50, 1) # hidden layer boyutu 50, output boyutu 1
        #self.sigmoid = nn.Sigmoid() # sigmoid aktivasyon fonksiyonu
    def tanh(self, x):
        return torch.div(torch.exp(x) - torch.exp(-x), torch.exp(x) + torch.exp(-x))

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

    def forward(self, x):
        x = self.fc1(x)
        x = self.tanh(x)
        x = self.fc2(x)
        x = self.sigmoid(x)
        return x

model = Net()
output=model(X)
print(output.data)
```

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:

```
torch.manual_seed(190401068)
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(3, 50) # input boyutu 3, hidden layer boyutu 50
        #self.tanh = nn.Tanh() # tanh aktivasyon fonksiyonu
        self.fc2 = nn.Linear(50, 1) # hidden layer boyutu 50, output boyutu 1
        #self.sigmoid = nn.Sigmoid() # sigmoid aktivasyon fonksiyonu
    def tanh(self, x):
```

```

    return torch.div(torch.exp(x) - torch.exp(-x), torch.exp(x) + torch.exp(-x))

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

def forward(self, x):
    x = self.fc1(x)
    x = self.tanh(x)
    x = self.fc2(x)
    x = self.sigmoid(x)
    return x
model = Net()
output=model(X)
print(output.data)

```

tensor([[0.5278], [0.5029]])

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/ahmetsalihaymelek/Yapay_sinir_aglari/blob/main/4_FeedForward.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 yaşamış. 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 uyarılmışlar. 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 şans 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, büyülu 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.

Prensese, kendisini iyileştiren veri setini de, gelecekte onların izinden gidecek bilgisayar mühendisi prensler ve prensesler başkalarına faydalı olabilecek yapay sinir ağları oluşturmayı öğretsinler 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.fc2 = nn.Linear(hidden_size1, hidden_size2)
        self.fc3 = nn.Linear(hidden_size2, output_size)
        self.relu = nn.ReLU()
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):

        out = self.fc1(x)
        out = self.relu(out)
        out = self.fc2(out)
        out = self.relu(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.

```
def early_stopping(model, val_score, best_val_loss, patience, patience_counter):
    stop_training = False
    if best_val_loss is None:
        patience_counter = 0
        best_val_loss = val_score
        torch.save(model.state_dict(), "checkpoint.pt")
    elif best_val_loss < val_score:
        patience_counter += 1
        print("Earlystopping Patience Counter", patience_counter)
        if patience_counter == patience:
            stop_training = True
```

```

    else:
        best_val_loss = val_score
        torch.save(model.state_dict(), "checkpoint.pt")
        patience_counter = 0

    return best_val_loss, patience_counter, stop_training

torch.manual_seed(190401068)
batch_size = 16
input_size=len(train_df.columns) - 1
hidden_size1=100
hidden_size2=50
output_size=2

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = MLP(input_size, hidden_size1, hidden_size2, output_size).to(device)

criterion=nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)
num_epochs = 50

#####5.2#####
# Train ve validation d n g leri
train_loss_list=[]
val_loss_list=[]
best_val_loss=None
patience=5
for epoch in range(num_epochs):
    # E itim modu
    model.train()
    train_loss = 0.0
    correct = 0
    total = 0

    for data, target in train_loader:

        # Verileri tens rlere d n t r n
        data, target = data.to(device), target.to(device)

        # Gradyanlar s f rlay n
        optimizer.zero_grad()

        # leri do ru hesaplama
        output = model(data)
        #target= torch.flatten(target)
        loss = criterion(output, target)

        # Kayb geriye do ru hesaplay n ve gradiyanlar hesaplay n
        loss.backward()
        optimizer.step()

        # statistikleri takip edin
        train_loss += loss.item() * data.size(0)
        _, predicted = torch.max(output.data, 1)
        total += target.size(0)

        correct += (predicted == target).sum().item()

    train_loss /= len(train_loader.dataset)
    train_loss_list.append(train_loss)
    train_acc = 100 * correct / total

    # De erlendirme modu
    model.eval()
    val_loss = 0.0

```



```

with torch.no_grad():
    correct = 0
    total = 0
    for data, target in val_loader:
        data, target = data.to(device), target.to(device)

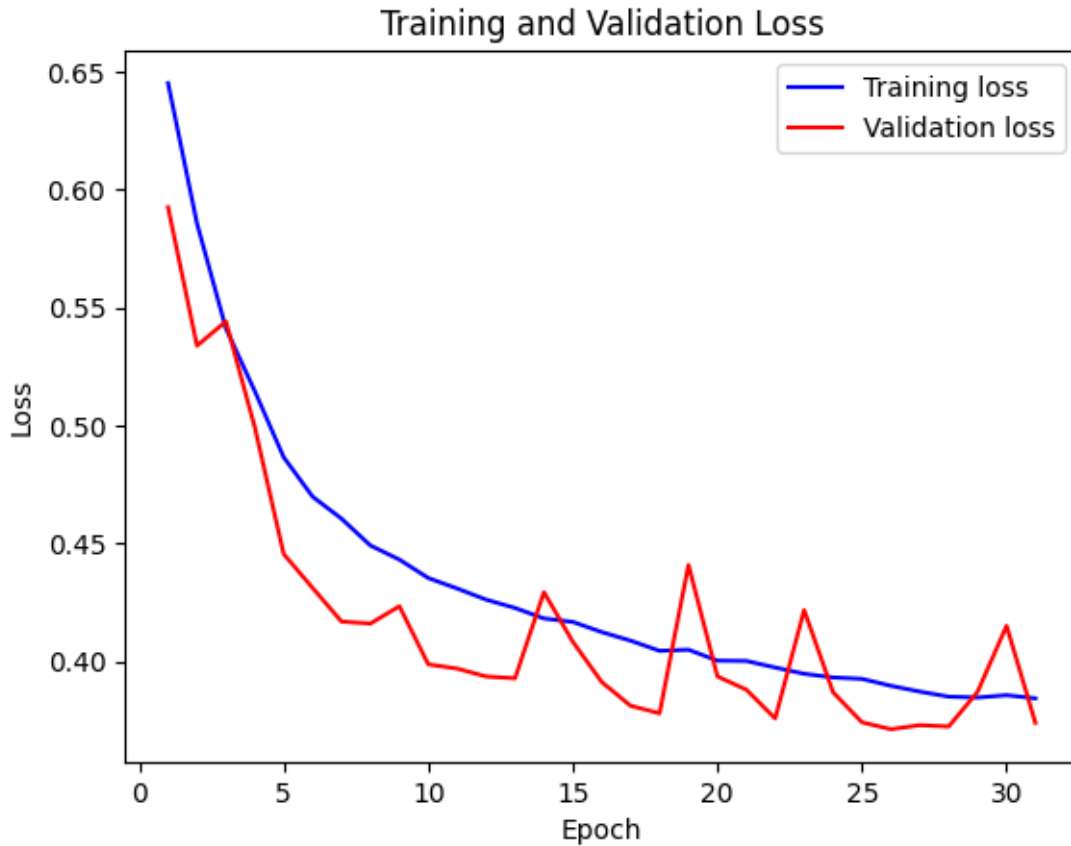
        # leri do ru hesaplama
        output = model(data)
        loss = criterion(output, target)

        # statistikleri takip edin
        val_loss += loss.item() * data.size(0)
        _, predicted = torch.max(output.data, 1)
        total += target.size(0)
        correct += (predicted == target).sum().item()

val_loss /= len(val_loader.dataset)
val_acc = 100 * (correct / total)
val_loss_list.append(val_loss)
val_score=val_loss
best_val_loss, patience_counter, stop_training=early_stopping(model,val_score,
                                                                best_val_loss,patience,
                                                                patience_counter)

if stop_training==True:
    break
#Her epok sonunda istatistikleri yazd r n
print(f'Epoch: {epoch+1}/{num_epochs} - Train Loss: {train_loss:.4f} - Train Acc:
      {train_acc:.2f}% - Val Loss: {
      val_loss:.4f} - Val Acc: {val_acc:.2f}
      %')

```



Şekil 1: Training and Validation Loss

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 dedim).

```
input_size=len(train_df.columns) - 1
hidden_size1=100
hidden_size2=50
output_size=2

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
#Farkl bir model ad yla olu turdumki alyormu g rebileyim
model2 = MLP(input_size, hidden_size1, hidden_size2, output_size).to(device)

#Buraa modelin i in a l klar koyduk

model2.load_state_dict(torch.load("/content/drive/MyDrive/yapay_sinir_aglari/
checkpoint.pt"))

from sklearn.metrics import confusion_matrix

# Modeli de erlendirme modunda ayarla
model.eval()

# Test seti zerinde d ng ye gir ve tahminler yap
with torch.no_grad():
    true_labels = []
    predicted_labels = []
    for inputs, labels in test_loader:
        inputs = inputs.to(device)
        labels = labels.to(device)
        outputs = model(inputs)
        _, predicted = torch.max(outputs.data, 1)
        true_labels += labels.cpu().numpy().tolist()
        predicted_labels += predicted.cpu().numpy().tolist()

# Confusion matrix'i hesapla
cm = confusion_matrix(true_labels, predicted_labels)

# Accuracy, precision, recall ve F1 skorlar n hesapla
accuracy = (cm.diagonal().sum()) / (cm.sum())
precision = (cm.diagonal() / cm.sum(axis=0)).mean()
recall = (cm.diagonal() / cm.sum(axis=1)).mean()
f1 = 2 * (precision * recall) / (precision + recall)
print("sadece model")
print(f"Test Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
print(f"F1 Score: {f1:.4f}")
```

Test Accuracy: 0.9288 Precision: 0.9288 Recall: 0.9287 F1 Score: 0.9288

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.

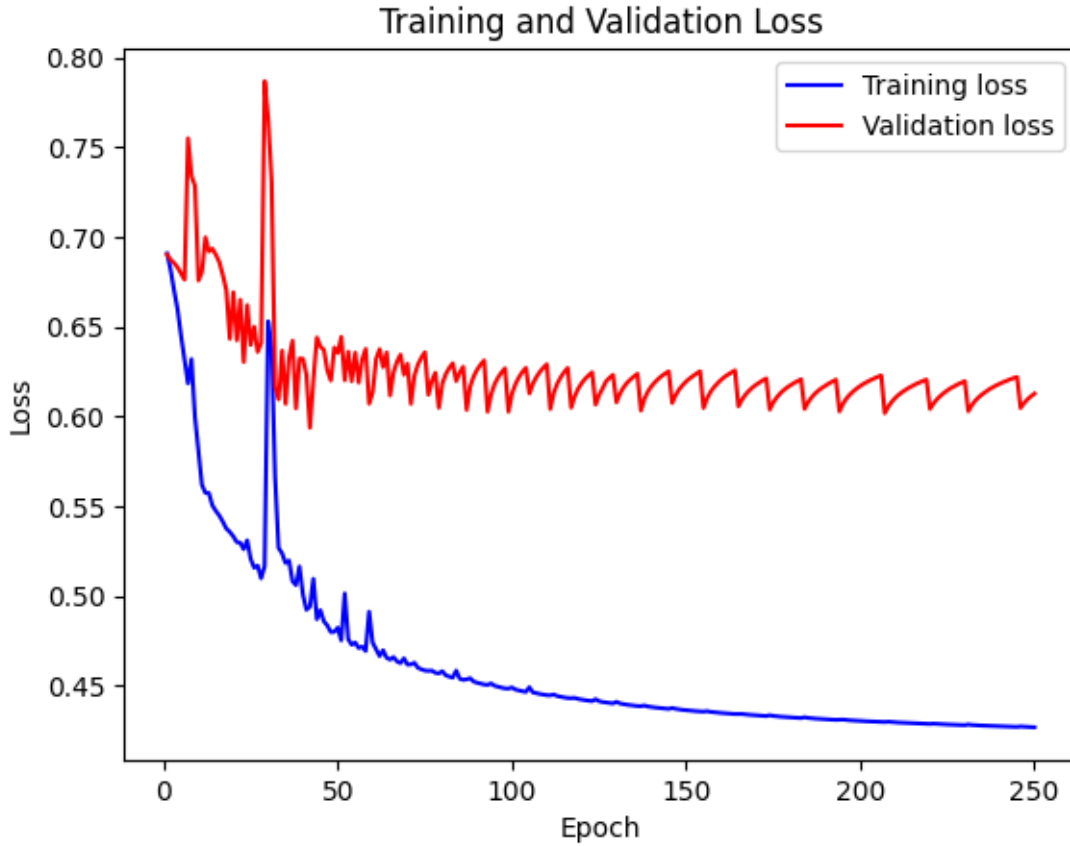
5.5 (3 Puan) Modelin eğitim setine overfit etmesi için elinizden geldiği kadar kodu gereken şekilde değiştirin, validasyon loss'unun açıkça yükselmeye başladığı, training ve

Tablo 1: Buraya bir açıklama yazın

Ortam	Süre (saniye)
CPU	04.60
GPU	15.12

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.

Cevap: learning reate ile oynadığımda training loss ile validation loss arasında çok fark olmadı modelin ne kadar hidden layeri ile oynasamda kötü sonuç çıkmadı tek çareyi modeli train ettiğimiz veriyi azaltmakta buldum sadece 500 tane veri verdim ve bu sonucu buldum



Şekil 2: overfit

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

https://github.com/ahmetsalihaymelek/Yapay_sinir_aglari/blob/main/5_MLP_Class.ipynb

6 (Toplam 10 Puan)

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

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

```
input_size=len(train_df.columns) - 1
hidden_size1=100
```

```

hidden_size2=50
output_size=2

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
#Farklı bir model adıyla oluşturdumki alyormu grebileyim
model = MLP(input_size, hidden_size1, hidden_size2, output_size).to(device)

criterion=nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)
num_epochs = 50

def early_stopping(model, val_score, best_val_loss, patience,patience_counter):
    stop_training = False
    if best_val_loss is None:
        patience_counter = 0
        best_val_loss = val_score
        torch.save(model.state_dict(), "checkpoint.pt")
    elif best_val_loss < val_score:
        patience_counter += 1
        print("Earlystopping Patience Counter", patience_counter)
        if patience_counter == patience:
            stop_training = True
    else:
        best_val_loss = val_score
        torch.save(model.state_dict(), "checkpoint.pt")
        patience_counter = 0

    return best_val_loss, patience_counter, stop_training

train_loss_list=[]
best_val_loss=None
patience=5
val_loss_list=[]
patience_counter = 0
#stop_training = False
for epoch in range(num_epochs):
    model.train()
    train_loss = 0.0
    correct = 0
    total = 0
    for data, target in train_loader:
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = criterion(output, target)
        #l1
        l1_lambda = 0.001
        l1_reg = torch.tensor(0.)
        for param in model.parameters():
            l1_reg += torch.norm(param, p=1)
        loss += l1_lambda * l1_reg
        loss.backward()
        optimizer.step()
        train_loss += loss.item() * data.size(0)
        _, predicted = torch.max(output.data, 1)
        total += target.size(0)
        correct += (predicted == target).sum().item()
    train_loss /= len(train_loader.dataset)
    train_loss_list.append(train_loss)
    train_acc = 100 * correct / total
    # De erlendirme modu
    model.eval()
    val_loss = 0.0
    with torch.no_grad():
        correct = 0

```

```

total = 0
for data, target in val_loader:
    data, target = data.to(device), target.to(device)
    output = model(data)
    loss = criterion(output, target)
    #l1
    l1_lambda = 0.001
    l1_reg = torch.tensor(0.)
    for param in model.parameters():
        l1_reg += torch.norm(param, p=1)
    loss += l1_lambda * l1_reg
    val_loss += loss.item() * data.size(0)
    _, predicted = torch.max(output.data, 1)
    total += target.size(0)
    correct += (predicted == target).sum().item()

val_loss /= len(val_loader.dataset)
val_acc = 100 * (correct / total)
val_loss_list.append(val_loss)
#EarlyStopping
val_score=val_loss
best_val_loss, patience_counter, stop_training=early_stopping(model, val_score,
                                                                best_val_loss, patience,
                                                                patience_counter)

if stop_training==True:
    break
print(f'Epoch: {epoch+1}/{num_epochs} - Train Loss: {train_loss:.4f} - Train Acc:
      {train_acc:.2f}% - Val Loss: {
      val_loss:.4f} - Val Acc: {val_acc:.2f}
      %')
```

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

L1 ve earlystopping sonrası Test Accuracy: 0.9456 Precision: 0.9481 Recall: 0.9458 F1 Score: 0.9470

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

L1 regülasyonu parametre seçiminde daha sert davranır. Modelimiz 90 accuracy değeri veriyordu bende daha yüksek olacağını düşündüğüm için L1 kullandım ikinci olarak earlystopping kullandım. Model belli bir süre öğrenmeyi bıraktı. Zaten yüksek oranda öğrendiği ve sürekli benzer loss larda seyrettiği için kullandım.

Earlystopping kullandım çünkü teste en iyi performans veren modeli sokmamız gerekir.

6.4 (1 puan) Sonucun github linkini aşağıya koyun:

https://github.com/ahmetsalihaymelek/Yapay_sinir_aglari/blob/main/6_Reg%C3%BC1%C3%BCzasyon.ipynb