

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: HATİCE SAIKE  
Öğrenci No: 180401042

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/haticesaike/neural\\_network\\_mid/blob/main/neural.tex](https://github.com/haticesaike/neural_network_mid/blob/main/neural.tex)

# 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 anladığınız terimleri de ChatGPT'ye sorup, sorduğunuz soruları ve ChatGPT'nin verdiği cevapları da sırayla aşağı koyun.**

**SORU:** Hopfield Network'leri nedir, hangi amaçla, nerede kullanılır? Hangi problemi çözer ve nasıl kullanılır, detaylı açıklayabilir misin?

**CEVAP:** Tabii, size yardımcı olmaktan mutluluk duyarım.

Hopfield ağları, yapay sinir ağı türlerinden biridir ve özellikle bellek işlevlerini simüle etmek için kullanılır. Bu ağlar, içindeki düğümlerin birbirleriyle bağlantıları aracılığıyla bilgi depolama ve hatırlama işlevlerini yerine getirirler.

Hopfield ağları, özellikle optimizasyon problemlerini çözmek için tasarlanmıştır. Optimizasyon problemleri, bir fonksiyonun minimum veya maksimum değerini bulma gibi matematiksel veya mühendislik problemleridir. Hopfield ağları, enerji fonksiyonları kullanarak bu optimizasyon problemlerini çözmek için kullanılır. Ağ, enerji fonksiyonunu minimize etmek için girdi verileri üzerinde çalışır.

Bir Hopfield ağı, bir matris şeklinde ifade edilebilir. Bu matris, ağın düğümleri arasındaki bağlantıları gösterir. Matrisin köşegenleri sıfırdır, çünkü bir düğüm kendisiyle doğrudan bağlantılı değildir. Ağın diğer bağlantıları simetrik olmalıdır, çünkü bağlantılar birbirine bağlıdır.

Hopfield ağları, verilerin depolanması, tanınması ve sıkıştırılması gibi birçok alanda kullanılır. Örneğin, Hopfield ağına bir dizi resim verilebilir ve ağ, benzer özelliklere sahip resimleri gruplayarak verileri sıkıştırabilir. Ayrıca, Hopfield ağı, ham verileri temizleyebilir veya eksik verileri tamamlayabilir.

Hopfield ağları, yapay sinir ağlarına giriş yapmak isteyenler için de bir başlangıç noktası olarak kullanılabilir. Bu ağlar, basit yapısı ve kolayca anlaşılabilirliği sayesinde yapay sinir ağları hakkında genel bir anlayış sağlamak için kullanılabilirler.

Hopfield ağlarının kullanımı, ağın nasıl eğitildiğine bağlıdır. Bir Hopfield ağı, ağın girdilerinin hepsinin aynı anda verilmesiyle eğitilir. Eğitim işlemi sırasında, ağ girdi verilerini depolar ve enerji fonksiyonunu minimize etmek için çalışır. Eğitim tamamlandığında, ağ verilen bir girdiye en yakın olan depolanan veriyi hatırlayabilir veya verileri sıkıştırabilir.

**Bellek işlevlerini simüle etmek derken neyi kastediyorsun açıklar mısın?**

Hopfield ağları, insan belleği gibi bilgi depolama ve hatırlama işlevlerini simüle etmek için kullanılır. Bu ağlar, öğrenme yoluyla veri örüntülerini depolayabilir ve daha sonra bu örüntüleri hatırlamak için kullanabilir. Ayrıca, veri örüntüleri arasındaki benzerlikleri tanıyabilir ve sınıflandırabilirler.

Hopfield ağları, birçok alanda bellek işlevlerini simüle etmek için kullanılabilir. Örneğin, birçok uygulama, Hopfield ağlarını görüntülerin tanınması, yüz tanıma, ses tanıma, doğal dil işleme ve finansal tahminler gibi alanlarda kullanır.

Özetle, Hopfield ağları, bellek işlevlerini simüle ederek, bir veri kümesindeki örüntüleri depolayabilir ve daha sonra bu örüntüleri hatırlayabilirler. Bu, birçok uygulamada kullanılabilen güçlü bir özelliktir.

**Hopfield ağlarının çözdüğü optimizasyon problemlerine bir örnek verir misin?**

Tabii, bir örnek vermek gerekirse, Hopfield ağları, seyahat eden satıcı problemi gibi NP-zorlu optimizasyon problemlerini çözmek için kullanılabilir. Seyahat eden satıcı problemi, bir satıcının verilen bir dizi şehre en kısa yolculuk mesafesiyle gitmesi gereken bir problem olarak tanımlanabilir. Bu problem, sayısal yöntemlerle çözülebilir ancak Hopfield ağları gibi alternatif yaklaşımlar da kullanılabilir. Hopfield ağları, seyahat eden satıcı problemi gibi problemleri enerji fonksiyonlarına dönüştürerek çözebilir. Problemin optimal çözümü, enerji fonksiyonunun minimum değerine karşılık gelir. Hopfield ağı, enerji fonksiyonunu minimize ederek optimal çözümü elde eder.

**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 ağı olarak isimlendirilir. İnsan zihnindeki bilgiler sinir ağları aracılığı ile iletilir ve yine aynı ağ aracılığı ile çeşitli bilgileri analiz edip, harmanlayarak bir cevap döndürür. Hopfield ağları insan zihninin bu işleyişinin simüle edilmiş hali gibi düşünülebilir. Hopfield ağları kendi içinde bir sürü bilgi ağı oluşturur ve bu ağların içindeki verileri bağlantılı olduğu diğer verileri de baz alarak yorumlar, böylece bir karar mekanizması oluşmuş olur. Birtakım bilgileri temel almasını söyleyerek yeni bilgileri önceki veriler ışığında yorumlamasını sağlıyoruz aslında. Örneğin eğitilmiş bir dataset verdiğimiz varsayalım, içinde people labelına sahip resimler yer alsın. Hopfield ağlarının da sayesinde bu bilgiler depolanır ve benzer bir resim gördüğünde people olup olmadığıyla ilgili yorum yapabilecek bir mekanizmaya sahip olur. Yani Hopfield ağları, birtakım verileri depolama, analiz etme ve gerektiğinde kullanabilmeyi sağlayan bir bilgi ağıdır, diyebiliriz. Kullanım alanı geniş olmakla beraber özellikle makine öğrenmesi konusunda hopfield ağlarından yararlanılabileceğini düşünüyorum. Ayrıca yüz tanıma, ses tanıma gibi programları incelediğimizde hopfield ağlarının gerçek hayatta oldukça etkin bir biçimde kullanıldığını görürüz

## 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 nedir, ne işe yarar, hangi ihtiyaçtan doğmuştur detaylı bir şekilde açıklayabilir misin?

**CEVAP:** Bir Boltzmann makinesi, yapay sinir ağı alanında bir tür denetimsiz öğrenme algoritmasıdır. Adını, istatistik mekaniği ve termodinamiği alanında çalışmalar yapmış olan Ludwig Boltzmann'dan almaktadır. Boltzmann makinesi, hem girdi hem de çıktı katmanına sahip olan bir yapay sinir ağıdır ve unsupervised learning (denetimsiz öğrenme) için tasarlanmıştır.

Bir Boltzmann makinesi, iki tür düğümden oluşur: gizli düğümler ve görünür düğümler. Gizli düğümler, ağı içinde bulunurlar ve diğer düğümlere bağlıdır. Görünür düğümler ise ağı girdi katmanını oluşturur ve dışarıdan verilen verileri temsil ederler. Bu veriler, ağı gizli düğümleri tarafından işlenir ve sonuç olarak ağı çıktılar elde edilir.

Boltzmann makinesi, belirli bir veri setini öğrenmek için kullanılır. Bu veri seti, ağı girdi katmanına verilir ve ağı gizli düğümleri, verilerin belirli özelliklerini öğrenmek için birbirleriyle etkileşime girerler. Bu özellikler, ağı öğrenme sürecinde otomatik olarak belirlenir ve veri setinin genel yapısını temsil eder.

Boltzmann makinesi, birçok alanda kullanılır. Özellikle, verilerin boyutunu azaltmak veya gizli yapıları ortaya çıkarmak için kullanılır. Örneğin, bir görüntü veri setindeki yüzleri tanımak için kullanılabilir. Ayrıca, doğal dil işleme ve sınıflandırma gibi diğer alanlarda da kullanılabilir.

Boltzmann makinesi, öğrenme sırasında enerji fonksiyonu olarak adlandırılan bir fonksiyonu minimize etmek için çalışır. Bu enerji fonksiyonu, ağı girdi verilerini ve ağı gizli düğümlerini kullanarak hesaplanır. Ağı öğrenme işlemi, enerji fonksiyonunun minimum değerini bulmak için gerçekleştirilir. Bu minimum değer, ağı verileri en iyi şekilde temsil edebilecek bir konfigürasyona sahip olduğunu gösterir.

Boltzmann makinesinin tasarlanması, yapay sinir ağlarının denetimsiz öğrenme ihtiyacından doğmuştur.

Denetimsiz öğrenme, özellikle büyük veri setleri üzerinde çalışırken verilerin birçoğunun etiketlenmemesi durumunda kullanışlıdır.

### Unsupervised learning nedir?

Unsupervised learning, makine öğrenmesi alanında, etiketlenmemiş verilerin kullanılarak yapılan öğrenme yöntemidir. Bu yöntemde, veri kümesindeki herhangi bir özellik veya sınıflandırma bilgisi verilmez ve model kendi kendine verilerdeki yapıları keşfetmeye çalışır.

Unsupervised learning, verilerin özelliklerini ve desenlerini keşfetmek için kullanılabilir. Bu, veri setindeki benzer özelliklere sahip örnekleri gruplandırma, benzer olmayan özellikleri olan örnekleri ayırma ve verilerin boyutunu azaltma gibi işlemleri içerebilir.

Bu yöntem, geniş veri setlerinde veri keşfi ve ön işleme için sıklıkla kullanılır. Ayrıca, kümeler arasındaki ilişkileri belirlemek veya verilerin gizli yapılarını öğrenmek için kullanılabilir.

Örnek olarak, bir restoranın müşterilerinden gelen sipariş verilerini ele alabiliriz. Sipariş verilerindeki herhangi bir sınıflandırma bilgisi olmadan, unsupervised learning yöntemleri kullanarak benzer ürünleri bir araya getirebilir ve bu bilgiyi restoranın menüsünde kullanabiliriz. Bu sayede müşterilerin ilgi alanlarını daha iyi anlayabilir ve onlara daha iyi bir deneyim sunabiliriz.

**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!**

Boltzman makinesi bir denetimsiz öğrenme algoritmasıdır. Gizli ve görünür düğümlerden oluşur. Görünür düğümler dışarıdan girilen verilerdir. Gizli düğümler görünür düğümlerden gelen bilgileri işleyerek bir çıktı oluşturur. Belirli bir dataseti öğrenmek için kullanılabilir. Dataset girdi katmanı olarak verilir ve gizli katman, girdi katmanında yer alan datasetin verilerini kullanarak belirli özellikleri öğrenmeye çalışır. Ancak datasetteki veriler etiketli değildir, denetimsiz öğrenme derken kastedilen de budur. Gizli katmanda belirli özelliklerden ortak paydalar çıkarılır ve veriler bu kümeleme sonucunda kategorize edilir. Boltzman makinesinde kullanılan ve analizin başarısını tespit etmeye yarayan enerji fonksiyonu vardır. Bu fonksiyonun değeri ne kadar küçük çıkarsa öğrenme o kadar başarılı sayılır. Süreç boyunca amaç enerji fonksiyonunu minimum düzeye çekmektir.

## 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.**

**SORU:** Markov assumption ve Markov Chain nedir, nerede kullanılır detaylı açıkla mısın?

**CEVAP:** Markov assumption, bir durumun gelecekteki davranışının yalnızca mevcut duruma bağlı olduğunu varsayar. Yani, bir sistemin gelecekteki davranışı, sadece şu anki durumuna bağlıdır ve geçmiş durumları veya olayları dikkate almaz. Bu varsayım, Markov Chain (Markov Zinciri) olarak adlandırılan bir olasılık modeli için temel bir varsayımdır.

Markov Chain, bir dizi durumun ardışık olarak olduğu ve her durumun yalnızca mevcut duruma bağlı olduğu bir olasılık sürecidir. Bu durumlar birbirine bağlıdır ve her bir durumun gerçekleşme olasılığı, yalnızca önceki durumun gerçekleşme olasılığına bağlıdır. Yani, Markov Chain'de, bir durumun gerçekleşme olasılığı, yalnızca önceki durumun gerçekleşme olasılığına bağlıdır ve diğer geçmiş durumlar veya olaylar göz önüne alınmaz.

Markov Chain, birçok alanda kullanılır. Örneğin, doğal dil işleme (NLP) ve makine çevirisi alanlarında sıklıkla kullanılır. Burada, bir kelimenin veya kelime dizisinin önceki kelimelere bağlı olarak olasılığı modellenir. Finansal piyasalar ve ekonomi gibi alanlarda da kullanılır. Burada, fiyat hareketlerinin ardışık durumları modellenir ve gelecekteki fiyatların tahmin edilmesine yardımcı olur. Ayrıca, zaman serileri analizi, rastgele yürüyüş modelleri, sinyal işleme ve kontrol sistemleri de Markov Chain kullanım alanlarıdır.

Markov Chain, özellikle belirli koşullar altında, istatistiksel olarak tahmin edilebilir bir yapıya sahiptir. Bu nedenle, Markov Chain, belirli bir sürecin öngörülebilirliğini ve modellenmesini sağlar. Bununla birlikte, Markov Chain'nin uygulanması için sistemin Markov assumption'a uygun olması gerekir, yani gelecekteki durumun yalnızca mevcut duruma bağlı olduğu varsayımı geçerli olmalıdır.

**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, gelecekteki bir durumun gerçekleşme olasılığının sadece bugünkü durum bilgisine bağlı olduğunu varsayar. Yani bir durumla ilgili olasılıklar geçmişteki durumlardan bağımsızdır, sadece bugünkü durum bilgisi önemlidir. Markov Chain ise Markov assumption'u temel alan bir matematiksel olasılık modelidir. Markov Chain'nin uygulanması için gelecekteki durumun yalnızca mevcut duruma bağlı olduğu varsayımı geçerli olmalıdır. Markov Chain'e göre birbirine bağlı olan olasılık durumlarından herbirinin gerçekleşme olasılığı sadece önceki durumun gerçekleşme olasılığına bağlıdır. Bu model birçok alanda kullanılır, örneğin makine çevirisi alanında bir kelime veya kelime grubunun önceki kelimelere bağlı olarak ne olabileceği tahmin edilir. Benzer mantıkla hava durumu tahmininde de kullanılır. Bir hava durumu tahmini yapmak için, bugünkü hava durumunu "durum" olarak kabul edilebilir. Daha sonra, geçiş olasılıkları, bugünkü durumun yarınki duruma geçiş olasılığını gösterir. Yani geçmişteki hava durumlarından bağımsız havanın bugünkü durumuna göre gelecekte olabilecek durumlar için olasılıklar oluşturulur.

#### 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
torch.manual_seed(1)
import torch.nn as nn

class Simplenet(nn.Module):
    def __init__(self, input_size=3, hidden_size=50):
        super(Simplenet, self).__init__()
        self.hidden_layer = nn.Sequential(
            nn.Linear(input_size, hidden_size),

        )
        self.output_layer = nn.Sequential(
            nn.Linear(hidden_size, 1),

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

    def my_sigmoid(self, x):
        return 1/ (1+ torch.exp(-x))
    def forward(self, x):
        x = self.hidden_layer(x)
        x = self.my_tanh(x)
        x = self.output_layer(x)
        x = self.my_sigmoid(x)
        return x

X = torch.tensor([[1, 2, 3], [4, 5, 6]], dtype=torch.float32)
model = Simplenet()
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:**

```
import torch
torch.manual_seed(180401042)
import torch.nn as nn

class Simplenet(nn.Module):
    def __init__(self, input_size=3, hidden_size=50):
        super(Simplenet, self).__init__()
        self.hidden_layer = nn.Sequential(
            nn.Linear(input_size, hidden_size),

        )
        self.output_layer = nn.Sequential(
            nn.Linear(hidden_size, 1),

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

    def my_sigmoid(self, x):
        return 1/ (1+ torch.exp(-x))
    def forward(self, x):
        x = self.hidden_layer(x)
        x = self.my_tanh(x)
        x = self.output_layer(x)
        x = self.my_sigmoid(x)
```

```

        return x

X = torch.tensor([[1, 2, 3], [4, 5, 6]], dtype=torch.float32)
model = SimpleNet()
output=model(X)
print(output.data)

```

tensor([[0.4695], [0.4501]])

**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/haticesaike/neural\\_network\\_mid/blob/main/4/Feed\\_Forward.ipynb](https://github.com/haticesaike/neural_network_mid/blob/main/4/Feed_Forward.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 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 öğretilsinler 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):
        super(MLP, self).__init__()
        self.fc1 = nn.Linear(13, 100)
        self.fc2 = nn.Linear(100, 50)
        self.fc3 = nn.Linear(50, 2)
        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

device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model = MLP().to(device)
```

**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 train(model, optimizer, criterion, train_loader, val_loader, num_epochs):

    best_loss = float('inf')
    for epoch in range(num_epochs):
        running_loss = 0.0
        running_corrects = 0
        total_train = 0

        model.train()

        for inputs, labels in train_loader:
            inputs, labels = inputs.to(device), labels.to(device)
            optimizer.zero_grad()

            # forward + backward + optimize
            outputs = model(inputs)
            loss = criterion(outputs, labels)
```



```

        loss.backward()
        optimizer.step()

        running_loss += loss.item()
        _, preds = torch.max(outputs, 1)
        running_corrects += torch.sum(preds == labels.data)
        total_train += labels.size(0)

    train_loss = running_loss / len(train_loader)
    train_acc = 100. * running_corrects / total_train
    train_losses.append(train_loss)
    train_accs.append(train_acc)

    # validation loss ve accuracy hesaplama
    val_loss, val_acc = evaluate(model, criterion, val_loader)
    val_losses.append(val_loss)
    val_accs.append(val_acc)

    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}")

def evaluate(model, criterion, data_loader):
    running_loss = 0.0
    running_corrects = 0
    total_val = 0

    model.eval()

    with torch.no_grad():
        for inputs, labels in data_loader:
            inputs, labels = inputs.to(device), labels.to(device)

            outputs = model(inputs.float())
            loss = criterion(outputs, labels)

            running_loss += loss.item()
            _, preds = torch.max(outputs, 1)
            running_corrects += torch.sum(preds == labels.data)
            total_val += labels.size(0)

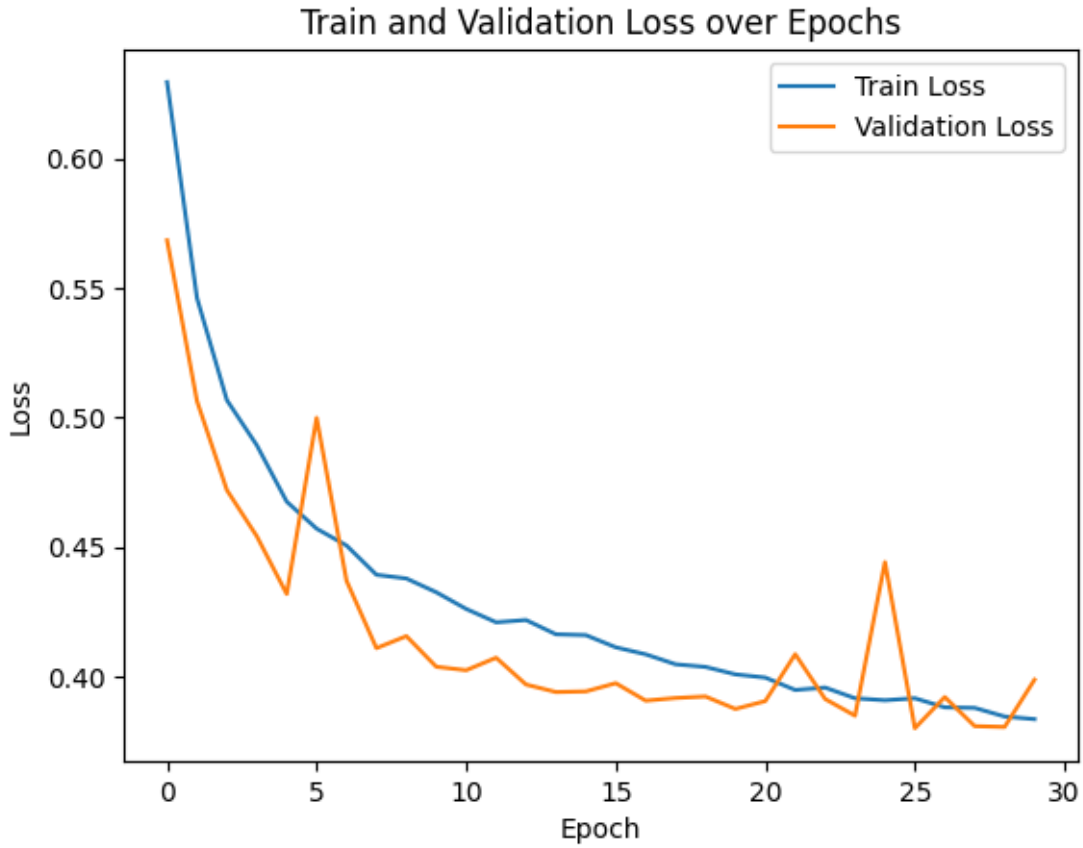
    val_loss = running_loss / len(data_loader)
    val_acc = 100. * running_corrects / total_val

    return val_loss, val_acc

torch.manual_seed(180401042)

train(model, optimizer, criterion, train_loader, validation_loader, num_epochs)

```



Şekil 1: Train and Validation Loss over Epochs

**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).**

```
from sklearn.metrics import accuracy_score, precision_score, f1_score, recall_score

def evaluate_metrics(model, criterion, data_loader):
    model.eval()
    y_true = []
    y_pred = []
    losses = []

    with torch.no_grad():

        for inputs, labels in data_loader:
            inputs, labels = inputs.to(device), labels.to(device)
            outputs = model(inputs)
            loss = criterion(outputs, labels)
            losses.append(loss.item())
            _, preds = torch.max(outputs, 1)
            y_true.extend(labels.cpu().numpy())
            y_pred.extend(preds.cpu().numpy())

    acc = accuracy_score(y_true, y_pred)
    prec = precision_score(y_true, y_pred, average='weighted')
    f1 = f1_score(y_true, y_pred, average='weighted')
    recall = recall_score(y_true, y_pred, average='weighted')
```

```
print(f"Test Accuracy: {acc:.4f},Prec Accuracy: {prec:.4f},F1 score: {f1:.4f},
      Recall score: {recall:.4f}")
```

```
evaluate_metrics(model, criterion, validation_loader)
```

Test Accuracy: 0.9268,Prec Accuracy: 0.9319,F1 score: 0.9266,Recall score: 0.9268

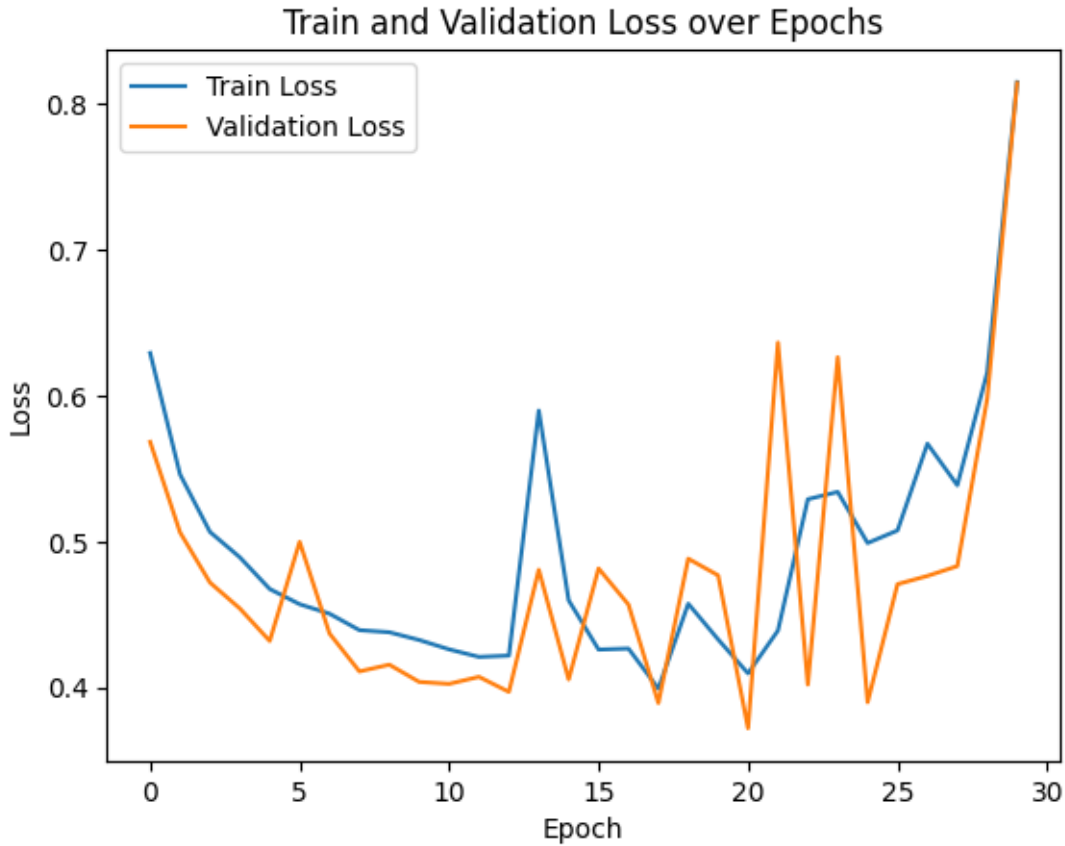
**5.4 (5 Puan)** Tüm kodların CPU’da çalışması ne kadar sürüyor hesaplayın. Sonra to device yöntemini kullanarak modeli ve verileri GPU’ya atıp kodu bir de böyle çalıştırın ve ne kadar sürdüğünü hesaplayın. Süreleri aşağıdaki tabloya koyun. GPU için Google Colab ya da Kaggle’ı kullanabilirsiniz, iki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

Tablo 1: Buraya bir açıklama yazın

Ortam	Süre (saniye)
CPU	0:00:14.163842
GPU	0:00:13.492997

**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.

Regularization kullanılmadı Her epochtan sonra val loss değeri kademeli bir şekilde artırıldı.



Şekil 2: Overfitting

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

[https://github.com/haticesaike/neural\\_network\\_mid/tree/main/5](https://github.com/haticesaike/neural_network_mid/tree/main/5)

## 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:

```
def train(model, optimizer, criterion, train_loader, val_loader, num_epochs):

    best_loss = float('inf')
    for epoch in range(num_epochs):
        running_loss = 0.0
        running_corrects = 0
        total_train = 0

        model.train()

        for inputs, labels in train_loader:
            inputs, labels = inputs.to(device), labels.to(device)
            optimizer.zero_grad()

            # forward + backward + optimize
            outputs = model(inputs)
            loss = criterion(outputs, labels)

            #l2 reg
            # l2_lambda = 0.001
            # l2_reg = torch.tensor(0., device=device)
            # for param in model.parameters():
            #     l2_reg += torch.norm(param)
            # loss += l2_lambda * l2_reg

            l1_lambda = 0.001
            l1_reg = torch.tensor(0., requires_grad=False, device=device)
            for param in model.parameters():
                l1_reg += torch.norm(param, p=1)
            loss += l1_lambda * l1_reg.float()
            loss.backward()
            optimizer.step()

            running_loss += loss.item()
            _, preds = torch.max(outputs, 1)
            running_corrects += torch.sum(preds == labels.data)
            total_train += labels.size(0)

        train_loss = running_loss / len(train_loader)
        train_acc = 100. * running_corrects / total_train
        train_losses.append(train_loss)
        train_accs.append(train_acc)

        val_loss, val_acc = evaluate(model, criterion, val_loader)
        val_losses.append(val_loss)
        val_accs.append(val_acc)

    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}")

if val_loss < best_loss:
    best_loss = val_loss
    best_model = model.state_dict()
    epochs_no_improve = 0
else:
    epochs_no_improve += 1
    if epochs_no_improve == 5:
        print("Early stopping!")
        break

def evaluate(model, criterion, data_loader):
    running_loss = 0.0
    running_corrects = 0
    total_val = 0

    model.eval()

    with torch.no_grad():
        for inputs, labels in data_loader:
            inputs, labels = inputs.to(device), labels.to(device)

            outputs = model(inputs.float())
            loss = criterion(outputs, labels)
            #l2 reg
            # l2_lambda = 0.001
            # l2_reg = torch.tensor(0., device=device)
            # for param in model.parameters():
            #     l2_reg += torch.norm(param)
            # loss += l2_lambda * l2_reg

            #l1 reg
            l1_lambda = 0.001
            l1_reg = torch.tensor(0., requires_grad=False, device=device)
            for param in model.parameters():
                l1_reg += torch.norm(param, p=1)
            loss += l1_lambda * l1_reg.float()

            running_loss += loss.item()
            _, preds = torch.max(outputs, 1)
            running_corrects += torch.sum(preds == labels.data)
            total_val += labels.size(0)

    val_loss = running_loss / len(data_loader)
    val_acc = 100. * running_corrects / total_val

    return val_loss, val_acc
train(model, optimizer, criterion, train_loader, validation_loader, num_epochs)

```

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

L1: Test Accuracy: 0.9204, Prec Accuracy: 0.9267, F1 score: 0.9201, Recall score: 0.9204

L2: Test Accuracy: 0.9268, Prec Accuracy: 0.9319, F1 score: 0.9266, Recall score: 0.9268

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

L1 ve L2 regülerizasyonu, makine öğrenmesinde aşırı uydurma (overfitting) sorununu çözmek için kullanılan tekniklerdir. Bu teknikler, modelin genellemesini artırarak aşırı uydurmaya neden olan parametreleri sınırlar.

L1 regülerizasyonu, Lasso olarak da bilinir ve parametreleri sıfıra yakınlaştırır. Bu, özellik seçimi yapar ve modeldeki gereksiz özelliklerin ağırlığını sıfıra indirerek daha basit bir model elde edilmesini sağlar.

L2 regülerizasyonu, Ridge olarak da bilinir ve parametreleri küçültür. Bu, parametreleri sıfıra yaklaştırmak yerine küçültür, böylece modelin genellemesi artar ve aşırı uydurma sorunu azalır. Ayrıca, L2 regülerizasyonu, çoklu doğrusal bağlantı (multicollinearity) gibi diğer sorunları da çözer.

L1 ve L2 regülerizasyonu, farklı özelliklere sahip veri kümelerinde farklı sonuçlar verebilir. Genel olarak, L2 regülerizasyonu daha sık kullanılır çünkü daha az agresif olduğu ve daha iyi sonuçlar verdiği düşünülür. Ancak, L1 regülerizasyonu, özellik seçimi için daha uygun olabilir.

Özetle, L1 ve L2 regülerizasyonu, aşırı uydurma sorununu çözmek için kullanılan faydalı tekniklerdir ve hangisinin kullanılacağı veri setinin özelliklerine bağlıdır.

L2 regülerizasyonu, parametreleri daha yumuşak bir şekilde kısıtladığı ve overfitting'e karşı daha iyi koruma sağladığı için daha iyi sonuç verdi.

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

[https://github.com/haticesaike/neural\\_network\\_mid/blob/main/6/MLP-Regularization.ipynb](https://github.com/haticesaike/neural_network_mid/blob/main/6/MLP-Regularization.ipynb)