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: Abdullah Sercan Sefunç
Öğrenci No: 190401027

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/sercansefunc/yapaysiniraglari

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.

Elman ağı ve Hopfield ağı gibi rekürrent ağlar arasında Hopfield Network'ler, özellikle bellek işleviyle ilgili uygulamalar için yaygın olarak kullanılan bir tür yapay sinir ağıdır.

Hopfield Network'ler, asenkron bir şekilde çalışır ve her iterasyonda her bir nöronun çıkışı güncellenir. Her nöronun, tüm diğer nöronlarla bağlantıları vardır ve nöronlar arasındaki bağlantılar ağırlıklarla belirlenir. Ağırlıklar, nöronların öğrenme sürecinde hesaplanır. Hopfield ağları, bir girdi vektörüne, belirli bir hedefi hatırlatmak için kullanılabilir. Bu, girdinin yetersiz veya bozulmuş olduğu durumlarda özellikle faydalıdır.

Hopfield Network'lerin öğrenme kuralı, Hebbian öğrenme kuralına dayanır. Yani, bir nöronun çıkışının diğer nöronların çıkışlarına olan bağlılığı, bu nöronların çıkışlarındaki değişimlere göre güncellenir. Ayrıca, Hopfield ağları, ağa bir hedef deseni yüklemek ve bu deseni bozulmuş bir şekilde sunmak için kullanılabilir. Bu durumda, ağ, hedef deseni hatırlamak için çıktılarını yeniden yapılandırır.

Hopfield Network'ler, özellikle bellek işlevleri için kullanılmalarına rağmen, sınıflandırma, çizgi tanıma ve veri sıkıştırma gibi diğer görevler için de kullanılabilirler. Ancak, Hopfield ağları, büyük veri kümeleriyle çalışmak için uygun değildir ve daha küçük boyutlu problemler için daha uygundur.

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 Network'ler, özellikle bellek işlevleri için kullanılan bir tür yapay sinir ağıdır. Her bir nöron, tüm diğer nöronlarla bağlantılıdır ve bu bağlantılar ağırlıklarla belirlenir. Bu ağırlıklar, öğrenme sürecinde hesaplanır. Ağ, bir girdi vektörüne belirli bir hedefi hatırlatmak için kullanılabilir ve girdi yetersiz veya bozulmuş olduğunda faydalı olabilir. Hopfield ağları, Hebbian öğrenme kuralına dayanır ve ağa bir hedef deseni yükleyerek bu deseni hatırlamak için çıktılarını yeniden yapılandırabilir. Bunun yanı sıra, sınıflandırma, çizgi tanıma ve veri sıkıştırma gibi diğer görevler için de kullanılabilirler. Ancak, büyük veri kümeleriyle çalışmak için uygun değildirler ve daha küçük boyutlu problemler için daha uygundurlar.

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 (BM), yapay sinir ağlarının bir türüdür ve genellikle öğrenme, özellikle de unsupervised (gözetimsiz) öğrenme için kullanılır. BM, hücrelerin veya nöronların birbirleriyle bağlantılı olduğu bir yapay sinir ağıdır ve enerji fonksiyonları üzerinden çalışır.

BM'nin temel bileşenleri gizli katman, gözlem katmanı ve bağlantı ağıdır. Gözlem katmanı, öğrenme için verilen girdiyi alırken, gizli katman ise girdiyi işler ve çıktıyı oluşturur. Bu katmanlar arasındaki bağlantılar, ağırlıklar ile belirlenir.

BM, birçok uygulama alanı bulur. Örneğin, BM, gizli yapıları (örneğin, yüzler veya diğer nesneler) keşfetmek için kullanılabilir. BM ayrıca, doğal dil işleme, öneri sistemleri, görüntü işleme, araştırma, sağlık sektörü ve daha birçok alanda kullanılabilir.

BM, ağın öğrenmesi sırasında, Gibbs dağılımı olarak adlandırılan bir istatistiksel yöntem kullanır. Bu yöntem, ağın enerji seviyelerini hesaplar ve ağırlıkları günceller. Ağın öğrenmesi sırasında, enerji fonksiyonu minimum değere yaklaşır ve ağırlıklar, öğrenme verileri üzerinde iteratif bir şekilde güncellenir.

BM, öğrenme verilerindeki düzensizlikleri ve belirsizlikleri ele almak için ideal bir araçtır. BM'nin birçok değişkeni olmasına rağmen, öğrenme verilerinin içindeki yapıları anlamak için etkili bir araçtır. Ancak, öğrenme süreci yavaş ve çıktıların yorumlanması zor olabilir. BM, özellikle büyük veri kümeleri için de uygun değildir.

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 (BM), yapay zeka alanında kullanılan bir ağ türüdür. Özellikle öğrenme işlemlerinde sıkça kullanılır. BM, birçok nöronun birbirleriyle bağlantılı olduğu bir yapıya sahiptir ve enerji seviyeleri üzerinden çalışır.BM'nin temel özellikleri arasında gizli ve gözlem katmanları ile bunları birbirine bağlayan ağırlıklar bulunur. Gözlem katmanı, veri girdilerini alırken, gizli katman ise bu verileri işleyerek çıktı üretir.BM, yüzler gibi gizli yapıların keşfedilmesi, doğal dil işleme, öneri sistemleri, görüntü işleme gibi alanlarda kullanılabilir.BM, öğrenme sürecinde enerji seviyelerini hesaplamak için Gibbs dağılımı adı verilen bir yöntem kullanır. Bu sayede, enerji fonksiyonu minimum değere yaklaşır ve ağırlıklar öğrenme verilerine göre güncellenir.BM, öğrenme verilerindeki düzensizlikleri ve belirsizlikleri ele almak için ideal bir araçtır. Ancak, öğrenme süreci biraz yavaş olabilir ve sonuçların yorumlanması zor olabilir. BM, büyük veri kümeleri için uygun olmayabilir.

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 varsayımı, bir sistemin gelecekteki durumunu sadece mevcut durumuna bağlı olarak tahmin edebileceğimizi belirten bir varsayımdır. Bu varsayım, herhangi bir zamandaki gelecekteki durumun, sadece mevcut durumla ilgili olduğunu ve geçmiş durumlarla ilgisi olmadığını ifade eder.

Markov Chain (Markov zinciri), Markov varsayımına dayalı bir matematiksel modeldir. Markov zinciri, bir dizi durumdan oluşur ve bir durumdan diğerine geçişlerin olasılıklarını belirleyen bir geçiş matrisi kullanır. Herhangi bir zamandaki durum, yalnızca önceki duruma bağlıdır ve önceki durumun geçmişi ile ilgisi yoktur.

Markov zinciri, pek çok alanda kullanılır. Örneğin, finansal modellerde, doğal dil işleme alanında, makine öğrenmesi ve yapay zeka algoritmalarında sıklıkla kullanılır. Özellikle zamanla değişen sistemleri modellemek için kullanılır. Markov zinciri, gelecekteki olayların tahmininde ve olasılıkların hesaplanmasında kullanılabilir.

Markov zinciri, birinci derece ve yüksek derece zincirler olmak üzere farklı tipleri vardır. Birinci derece zincirler, yalnızca bir önceki duruma bağlıdırken, yüksek derece zincirler, daha önceki durumların bir kombinasyonuna bağlıdır. Zincirlerin daha yüksek dereceleri, daha fazla geçmiş veriyi dikkate aldığından daha karmaşık olabilir.

Markov zinciri, ayrıca Monte Carlo Simülasyonu'nda ve bazı optimizasyon problemlerinde kullanılır. Ancak, zincirin uzunluğu arttıkça, hesaplama güçlüğü ortaya çıkabilir.

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 durumuna bağlı olarak tahmin edilebileceğini söyler. Markov zinciri ise bu varsayıma dayalı bir matematiksel modeldir. Bu model, bir dizi durumdan oluşur ve durumlar arasındaki geçiş olasılıklarını belirleyen bir matris kullanır. Markov zinciri, finansal modeller, doğal dil işleme, makine öğrenmesi ve yapay zeka gibi pek çok alanda kullanılır. Zincirlerin dereceleri farklı olabilir ve zincirin uzunluğu arttıkça hesaplama güçleşebilir. Ancak, Markov zinciri, gelecekteki olayların tahmininde ve olasılıkların hesaplanmasında oldukça faydalı bir araçtır.

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 numpy as np

torch.manual_seed(1)

X = np.array([[1, 2, 3], [4, 5, 6]], dtype=np.float32)
X_tensor = torch.from_numpy(X)

hidden_layer = torch.nn.Linear(3, 50)
activation = torch.nn.Tanh()
output_layer = torch.nn.Linear(50, 1)
output_activation = torch.nn.Sigmoid()
hidden = activation(hidden_layer(X_tensor))
output = output_activation(output_layer(hidden))
def forward(X):
```

```
    hidden = activation(hidden_layer(X))
    output = output_activation(output_layer(hidden))
    return output

output = forward(X_tensor)
print(output)
```

tensor([[0.6139], [0.6247]], gradfn ¡= Sigmoid

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

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

torch.manual_seed(190401027)

input_size = X.shape[1]
hidden_size = 50
W1 = torch.randn(input_size, hidden_size, dtype=torch.float)
b1 = torch.randn(hidden_size, dtype=torch.float)

output_size = 1
W2 = torch.randn(hidden_size, output_size, dtype=torch.float)
b2 = torch.randn(output_size, dtype=torch.float)

z1 = torch.matmul(X, W1) + b1
a1 = torch.tanh(z1)
z2 = torch.matmul(a1, W2) + b2
a2 = torch.sigmoid(z2)

print("a2:", a2)
```

a2: tensor([[0.3092], [0.2301]])

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/sercansefunc/yapaysiniraglari

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.

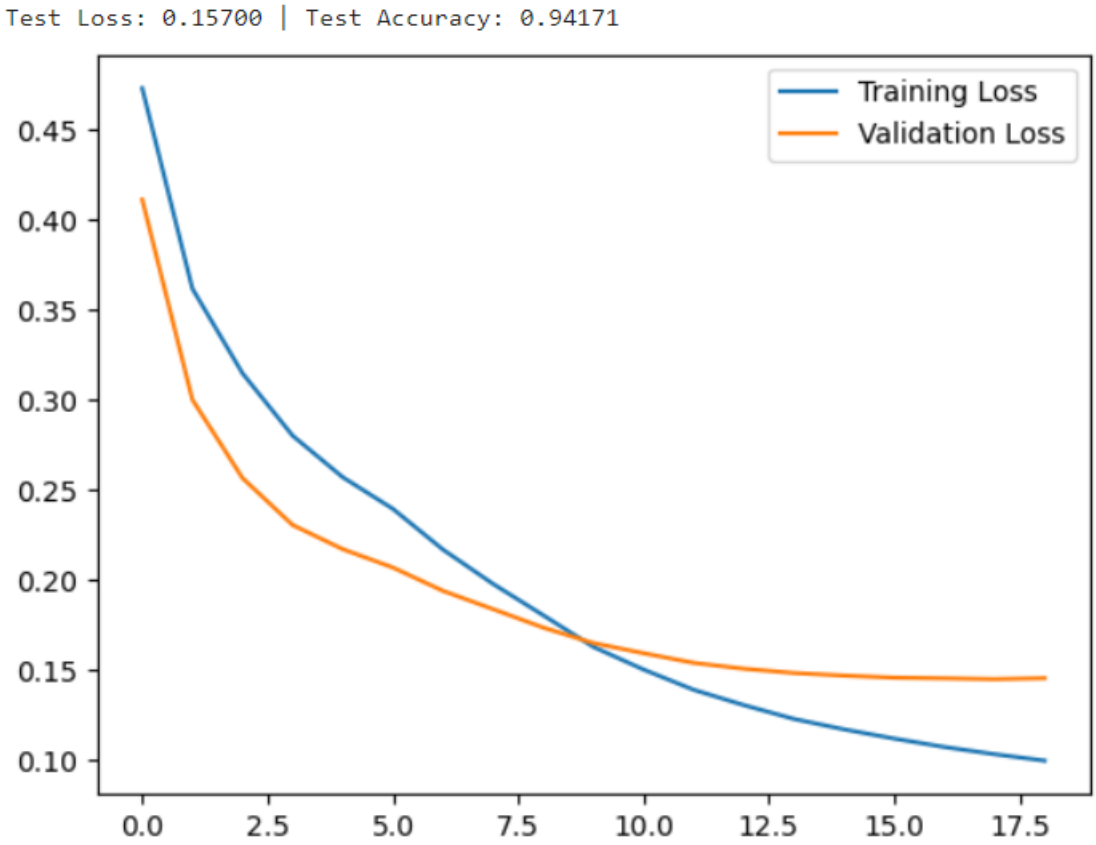
5.1 (10 Puan) Bu MLP'nin pytorch ile yazılmış class'ının kodunu aşağı kod bloğuna yapıştırın:

```
class MLP(nn.Module):
    def __init__(self):
        super().__init__()
        self.hidden1 = nn.Linear(13, 100)
        self.hidden2 = nn.Linear(100, 50)
        self.out = nn.Linear(50, 1)

    def forward(self, x):
        x = nn.functional.relu(self.hidden1(x))
        x = nn.functional.relu(self.hidden2(x))
        x = torch.sigmoid(self.out(x))
        return x
```

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.

```
model = MLP()
optimizer = optim.SGD(model.parameters(), lr=0.01)
criterion = nn.BCELoss()
torch.manual_seed(190401027)

# Train model
train_losses, val_losses = [], []
num_epochs = 1000
batch_size = 16

for epoch in range(num_epochs):
    for i in range(0, len(train_inputs), batch_size):
        batch_inputs = train_inputs[i:i+batch_size]
        batch_targets = train_targets[i:i+batch_size]

        optimizer.zero_grad()
        outputs = model(batch_inputs).squeeze()
        loss = criterion(outputs, batch_targets)
        loss.backward()
        optimizer.step()

    with torch.no_grad():
        train_outputs = model(train_inputs).squeeze()
        train_loss = criterion(train_outputs, train_targets)
        train_losses.append(train_loss.item())

        val_outputs = model(val_inputs).squeeze()
        val_loss = criterion(val_outputs, val_targets)
        val_losses.append(val_loss.item())

        train_preds = (model(train_inputs) > 0.5).float()
        test_preds = (model(test_inputs) > 0.5).float()
        val_preds = (model(val_inputs) > 0.5).float()

    if epoch > 10 and val_losses[-1] > val_losses[-3]:
        break

    if epoch % 1 == 0:
        print(f"Epoch {epoch} | Train Loss: {train_loss:.5f} | Val Loss: {val_loss:.
                                                    5f}")

test_outputs = model(test_inputs).squeeze()
test_loss = criterion(test_outputs, test_targets)
test_acc = ((test_outputs > 0.5).float() == test_targets).float().mean()

print(f"Test Loss: {test_loss:.5f} | Test Accuracy: {test_acc:.5f}")

plt.plot(train_losses, label="Training Loss")
plt.plot(val_losses, label="Validation Loss")
plt.legend()
plt.show()
```

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).



Şekil 1: Grafik

```
from sklearn.metrics import f1_score as f1, accuracy_score as acc, recall_score as
                                        rec, precision_score as prec

train_f1 = f1(train_targets, train_predictions)
train_acc = acc(train_targets, train_predictions)
train_recall = rec(train_targets, train_predictions)
train_precision = prec(train_targets, train_predictions)

test_f1 = f1(test_targets, test_predictions)
test_acc = acc(test_targets, test_predictions)
test_recall = rec(test_targets, test_predictions)
test_precision = prec(test_targets, test_predictions)

val_f1 = f1(val_targets, val_predictions)
val_acc = acc(val_targets, val_predictions)
val_recall = rec(val_targets, val_predictions)
val_precision = prec(val_targets, val_predictions)

print("Training set:")
print(f"F1 score: {train_f1:.5f} | Accuracy: {train_acc:.5f} | Recall: {train_recall
                                        :.5f} | Precision: {train_precision:.5f}")
print("Test set:")
print(f"F1 score: {test_f1:.5f} | Accuracy: {test_acc:.5f} | Recall: {test_recall:.5f
                                        } | Precision: {test_precision:.5f}")
print("Validation set:")
print(f"F1 score: {val_f1:.5f} | Accuracy: {val_acc:.5f} | Recall: {val_recall:.5f} |
                                         Precision: {val_precision:.5f}")
```

```
Training set:
F1 score: 0.97180 | Accuracy: 0.97204 | Recall: 0.97731 | Precision: 0.96635
Test set:
F1 score: 0.95251 | Accuracy: 0.95337 | Recall: 0.93041 | Precision: 0.97568
Validation set:
F1 score: 0.94937 | Accuracy: 0.94904 | Recall: 0.96154 | Precision: 0.93750
```

Şekil 2: Sonuçlar

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	kaç?
GPU	kaç?

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/sercansefunc/yapaysiniraglari

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:

```python
kod_buraya = None
if kod_buraya:
    devam_ise_buraya = 0

print(devam_ise_buraya)
```

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

Sonuçlar buraya.

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

Yorumlar buraya.

6.4 (1 puan) Sonucun github linkini aşağıya koyun:

www.benimgithublinkim2.com