



T.C

SAKARYA ÜNİVERSİTESİ

BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BSM 310 – YAPAY ZEKA

Grup üyeleri:

G171210045 Fatih Aslan

G171210035 Buğra Tuğrul

G161210083 Mehmet Büyükarıkan

Sakarya

2020

Yinelenen Sinir Ağları - İçindekiler

1-) Yapay Sinir Ağları

1.1) Yapay Sinir Ağları Tarihçesi

1.2) Yapay Sinir Ağları Nedir

2-) Derin Öğrenme

2.1) Derin Öğrenme Tarihçesi

2.2) Derin Öğrenme Nedir

2.3) Derin Öğrenme Mimarileri

3-) Yinelenen Sinir Ağları Tarihçesi

4-) Yinelenen Sinir Ağları Nedir

4.1) Yinelenen Sinir Ağları Mimarisi

4.2-) Yinelenen Sinir Ağları Nasıl Çalışır

5-) Yinelenen Sinir Ağları Tipleri

6-) Yinelenen Sinir Ağları Kullanılan Yerler

7-) Yinelenen Sinir Ağları Uzantıları

7.1) Çift Yönlü RNN

7.2) LSTM

7.3) Elman ve Jordan Ağları

7.4) Kapılı Yinelenen Birim

8-) Yinelenen Sinir Ağları Avantajlar ve Dezavantajlar

9-) Yinelenen Sinir Ağları Problemleri

9.1) Kaybolan Gradyan Problemi

9.2) Patlayan Gradyan Problemi

1-) Yapay Sinir Ağları

1.1) Yapay Sinir Ağları Tarihçesi

İlk yapay sinir ağı modeli 1943 yılında, bir sinir hekimi olan Warren McCulloch ile bir matematikçi olan Walter Pitts tarafından gerçekleştirilmiştir. McCulloch ve Pitts, insan beyninin hesaplama yeteneğinden esinlenerek, elektrik devreleriyle basit bir sinir ağı modellemişlerdir.

1.2) Yapay Sinir Ağları Nedir

Yapay sinir ağları, insan beyninin özelliklerinden olan öğrenme yolu ile yeni bilgiler türetebilme, oluşturabilme ve keşfedebilme gibi yetenekleri, herhangi bir yardım almadan otomatik olarak gerçekleştirebilmek amacı ile geliştirilen bilgisayar sistemleridir.

Yapay sinir ağları; insan beyninden esinlenerek, öğrenme sürecinin matematiksel olarak modellenmesi uğraşı sonucu ortaya çıkmıştır. Bu nedenledir ki, bu konu üzerindeki çalışmalar ilk olarak beyni oluşturan biyolojik üniteler olan nöronların modellenmesi ve bilgisayar sistemlerinde uygulanması ile başlamış, daha sonraları bilgisayar sistemlerinin gelişimine de paralel olarak bir çok alanda kullanılır hale gelmiştir.

2-) Derin Öğrenme

2.1) Derin Öğrenme Tarihçesi

Derin öğrenme, yapay sinir ağları ve insan beyninin işlevlerini taklit eden hesaplama sistemleri kavramına dayanır. Derin öğrenmenin tarihi, Warren McCulloch ve Walter Pitts'in 1943 yılında düşünce sürecini taklit etmek için matematiğe ve sinir mantığı olarak adlandırılan algoritmalara dayalı sinir ağları için bir hesaplama modeli oluşturmalarına uzanmaktadır. (McCulloch, Pitts, 1943).

2.2) Derin Öğrenme Nedir

Bugüne kadar derin öğrenmenin, literatürde farklı kaynaklarda değişik birçok tanımı yapılmıştır. Derin öğrenme; bilgisayarların, deneyimlerden öğrenmelerini ve dünyayı kavramların hiyerarşisi açısından anlamalarını sağlayan bir makine öğrenimi olarak tanımlanmıştır (Gu, Zhang, Kim, 2016). Başka bir kaynakta derin öğrenme; denetimli veya denetimsiz özellik çıkarma, dönüştürme, desen analizi ve sınıflandırma için birçok doğrusal olmayan gizli katmandan yararlanan bir makine öğrenme teknikleri sınıfı olarak tanımlanmıştır (Deng, Yu, 2014). Yine derin öğrenme; insan beyninin son derece karmaşık problemler için gözlemlene, analiz etme, öğrenme ve karar verme yeteneğini taklit etmeyi amaçlayan, büyük miktarda denetimsiz veri kullanan bir makine öğrenmesi olarak tanımlanmıştır (Najafabadi, Villanustre, Khoshgoftaar, Seliya, Wald, Muharemagic, 2015).

Derin öğrenme konusunda farklı kaynaklardan alınan tanımlar bileştirilecek olursa; derin öğrenme insan beyninin karmaşık problemler için gözlemlene, analiz etme, öğrenme ve karar verme gibi yeteneklerini taklit eden, denetimli veya denetimsiz olarak özellik çıkarma, dönüştürme ve sınıflandırma gibi işlemleri büyük miktarlardaki verilerden yararlanarak yapabilen bir makine öğrenmesi tekniğidir.

2.3) Derin Öğrenme Mimarileri

- Konvolüsyonel Sinir Ağları
- **Yinelenen Sinir Ağı (RNN)**
- Uzun Kısa Vadeli Memory
- Kısıtlanmış(Derin) Boltzmann Makineleri
- Derin İnanç Ağları
- Derin Oto-Kodlayıcılar

3-) Yinelenen Sinir Ağları Tarihçesi

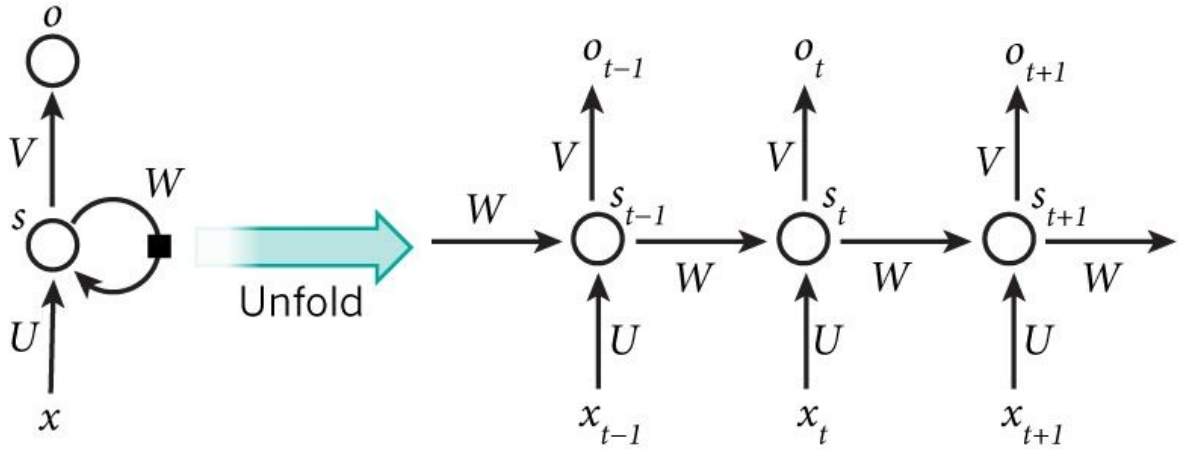
Yinelenen Sinir Ağları (YSA), ilk olarak John Hopfield tarafından 1982de keşfedildi, yazı ve ses gibi sıralı işlemlerde kullanıldı. Sinir ağlarının zamanla videolardaki hareketler ve ses kayıtlarındaki konuşmalar gibi kalıpları öğrenmesini sağladı.

4-) Yinelenen Sinir Ağları Nedir

RNN'ler olarak da bilinen tekrarlayan sinir ağları, gizli durumlara sahipken önceki çıktıların girdi olarak kullanılmasına izin veren bir sinir ağları sınıfıdır. Yani RNN'lerin arkasındaki fikir sıralı bilgi kullanmaktır. Geleneksel bir sinir ağında tüm girdilerin (ve çıktıların) birbirinden bağımsız olduğunu varsayabiliriz. Ancak birçok görev için bu çok kötü bir fikir. Bir cümledeki bir sonraki kelimeyi tahmin etmek isterseniz, hangi kelimelerin daha önce geldiğini daha iyi bilirsiniz. RNN'ler tekrar tekrar çağrılır, çünkü bir dizinin her elemanı için aynı görevi yaparlar, çıktı önceki hesaplamalara bağlıdır.

4.1) Yinelenen Sinir Ağları Mimarisi

Tipik olarak aşağıdaki gibidirler.



Tekrarlayan bir nöral ağ ve hesaplamanın zaman içinde ortaya çıkması, ileri hesaplamada yer alır.

Yukarıdaki diyagram, tam bir ağı açılmış olan bir RNN'yi göstermektedir. Kaydettiğimizde, tüm dizinin ağını yazdığımız anlamına gelir. Örneğin, umduğumuz sekans 5 kelimelik bir cümle ise, ağ her bir kelime için 5 katmanlı bir sinir ağına açılacaktır.

4.2-) Yinelenen Sinir Ağları Nasıl Çalışır

YSA'lar kendi çıkışlarından girişlerine bağlantı kurarak sinir ağlarının geçici bir hafıza kazanabilmesini sağlarlar. Bu tüm sinir ağını kopyalayıp aynı mimariyi yeni bir yapıya eklemek gibi düşünülebilir böylece yük hafifler. Örneğin, bir cümle içinde bir sonraki kelimeyi tahmin etmek için, o anki kelimedenden önce hangi sözcüklerin geldiğini bilmek gerekmektedir. RNN mimarisinin yinelenen (recurrent) olarak adlandırılmasının sebebi, bir dizinin her ögesi için (cümledeki kelimeler gibi) aynı görevi önceki çıktılarına bağlı olarak yerine getirmesidir.

Bir RNN'de hesaplamayı yöneten formüller aşağıdaki gibidir:

x_t , t adımında girdi. Örneğin, x_1 , bir cümlenin ikinci kelimesine karşılık gelen tek-sıcak bir vektör olabilir.

s_t , t adımında gizli durumdur. Bu, ağın "hafızası". s_t önceki gizli duruma ve mevcut adımdaki girdiye göre hesaplanır: $s_t = f(Ux_t + Ws_{t-1})$. F fonksiyonu genellikle tanh veya ReLU gibi doğrusal olmayan bir durumdur. İlk gizli durumu hesaplamak için gereken s_{-1} , tipik olarak tüm sıfırlara başlatılır.

o_t t adımındaki çıktıdır. Örneğin, bir cümledeki bir sonraki kelimeyi tahmin etmek isteseydik, kelime dağılımımız boyunca olasılıkların bir vektörü olurdu. $o_t = \text{softmax}(Vs_t)$.

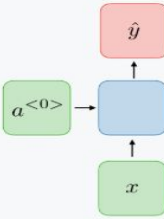
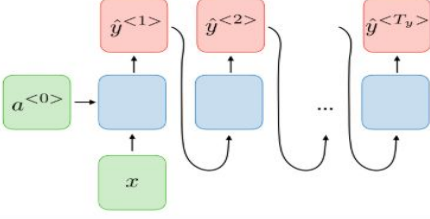
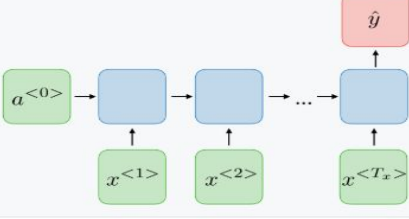
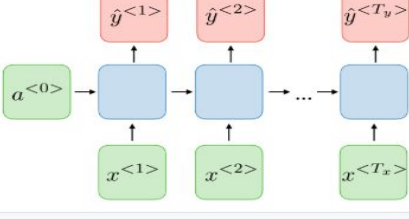
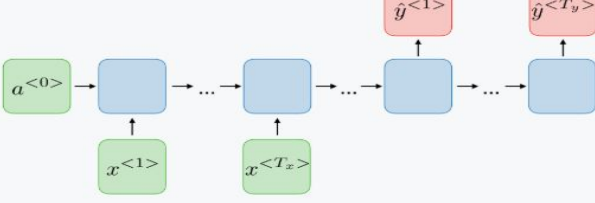
Burada dikkat edilmesi gereken birkaç nokta var:

Gizli durum s_t 'yi ağın belleği olarak düşünebilirsiniz. s_t , önceki zaman adımlarında ne olduğu hakkında bilgi yakalar. o_t adımındaki çıktı sadece t zamanında belleğe dayanarak hesaplanır. Yukarıda kısaca bahsedildiği gibi, uygulamada biraz daha karmaşıktır, çünkü s_t genellikle çok fazla zaman adımından gelen bilgileri yakalayamaz.

Her katmanda farklı parametreler kullanan geleneksel derin sinir ağından farklı olarak, bir RNN tüm adımlarda aynı parametreleri paylaşır. Bu, her adımda aynı görevi gerçekleştirdiğimiz gerçeğini, sadece farklı girdilerle yansıtır. Bu, öğrenmemiz gereken toplam parametre sayısını büyük ölçüde azaltır.

RNN'nin her zaman adımında çıktıları vardır, ancak göreve bağlı olarak bu gerekli olmayabilir. Örneğin, bir cümledeki duygularını kestirirken, her bir kelimedeki sonraki duyguları değil, sadece son çıktıyı önemseyebiliriz. Benzer şekilde, her zaman adımında girişlere ihtiyaç duymayabiliriz. Bir RNN'nin ana özelliği, bir dizi hakkında bazı bilgileri toplayan gizli durumudur.

5-) Yinelenen Sinir Ağları Tipleri

RNN Türü	Örnekleme	Örnek
Bire bir $T_x = T_y = 1$		Geleneksel sinir ağı
Bire çok $T_x = 1, T_y > 1$		Müzik üretimi
Çoka bir $T_x > 1, T_y = 1$		Duygu sınıflandırma
Çoka çok $T_x = T_y$		İsim varlık tanıma
Çoka çok $T_x \neq T_y$		Makine çevirisi

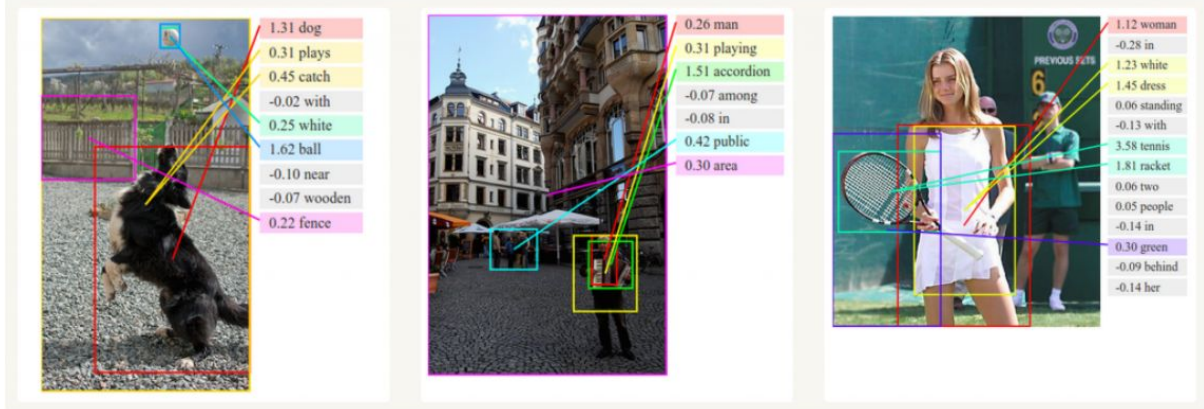
6-) Yinelenen Sinir Ağları Kullanılan Yerler

- Makine çevirisi
- Robot kontrolü
- Zaman serisi tahmini
- Konuşma tanıma
- Zaman serisi sapma tespiti
- Ritim öğrenme
- Müzik kompozisyonu
- Gramer öğrenme
- El yazısı tanıma
- İnsan eylem tanıma
- Protein homoloji tespiti
- Proteinlerin hücre içi lokalizasyonu tahmin
- İş süreci yönetimi konusunda çeşitli tahmin görevleri
- Tıbbi bakım yollarında tahmin

Yapılan Çalışmalar;

Alex Graves'in çalışmasında, ses verilerinin fonetik sunumuna gerek kalmadan doğrudan metne çeviren bir RNN tabanlı konuşma tanıma sistemi sunulmuştur.

RNN'ler CNN ile birlikte, etiketlenmemiş görüntüler için tanımlayıcı üreten bir modelin parçası olarak kullanılmıştır. Birleştirilmiş model, görüntüdeki nesneleri tanımlamanın yanında, tanımlayıcıların görüntülerdeki konumlarını bile bulmayı başarmıştır.

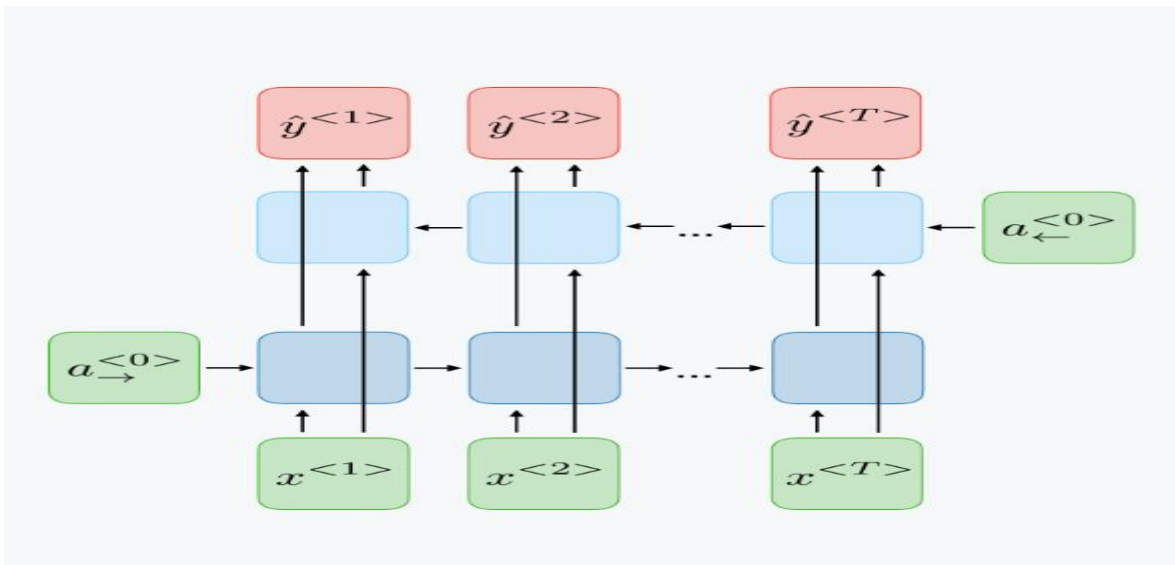


7-) Yinelenen Sinir Ağları Uzantıları

7.1) Çift Yönlü RNN

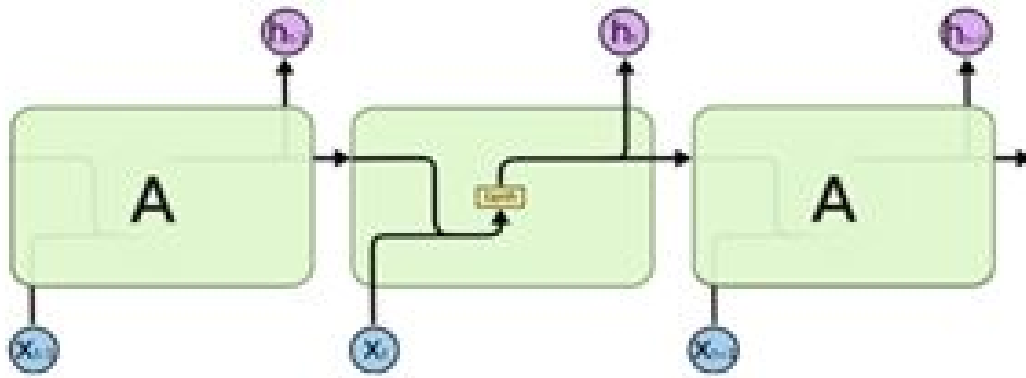
Çift yönlü RNN'ler, t 'deki çıkışın sadece dizideki önceki öğelere değil, aynı zamanda gelecekteki öğelere de bağlı olabileceği fikrine dayanır. Örneğin, bir sıradaki eksik kelimeyi tahmin etmek için hem sol hem de sağ içeriğe bakmak istiyorsanız Çift yönlü RNN'ler oldukça basittir. Birbirlerinin üzerine yığılmış iki RNN vardır. Daha sonra çıkış, her iki RNN'nin gizli durumuna göre hesaplanır.

Pratikte bu bize daha yüksek bir öğrenme kapasitesi sağlar (ama aynı zamanda çok fazla eğitim verisine de ihtiyacımız var).



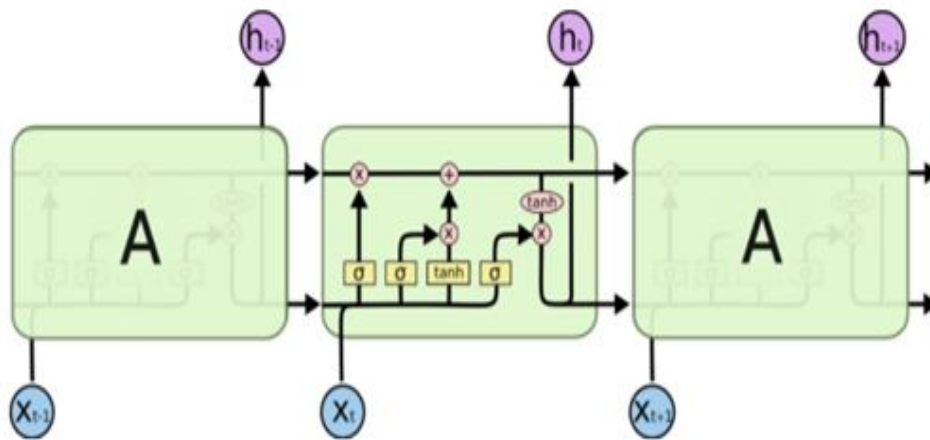
7.2) LSTM

Uzun Kısa Vadeli Hafıza Ağları, genellikle “LSTM’ler” olarak adlandırılır uzun vadeli bağımlılıkları öğrenebilen özel bir RNN türüdür. Bunlar Hochreiter & Schmidhuber (1997) tarafından tanıtıldı. Çok çeşitli sorunlar üzerine muazzam derecede çalışırlar ve şu anda yaygın olarak kullanılmaktadırlar. LSTM’ler, uzun vadeli bağımlılık sorununun önüne geçmek için tasarlanmıştır. Uzun süre bilgi hatırlamak pratikte varsayılan davranışlarıdır, öğrenmek için uğraştıkları bir şey değildir. Tüm tekrarlayan sinir ağları, tekrar eden sinir ağı modül zinciri biçimindedir. Standart RNN’lerde, bu yinelenen modül, tek bir tanh katmanı gibi çok basit bir yapıya sahip olacaktır.



The repeating module in a standard RNN contains a single layer.

LSTM’lerin de art arda birbiri takip eden yapıları vardır. Ancak bir sonraki parça farklı bir yapıya sahiptir. Tek bir sinir ağı katmanı yerine, çok özel bir şekilde etkileşimde olan dört parça var.



The repeating module in an LSTM contains four interacting layers.

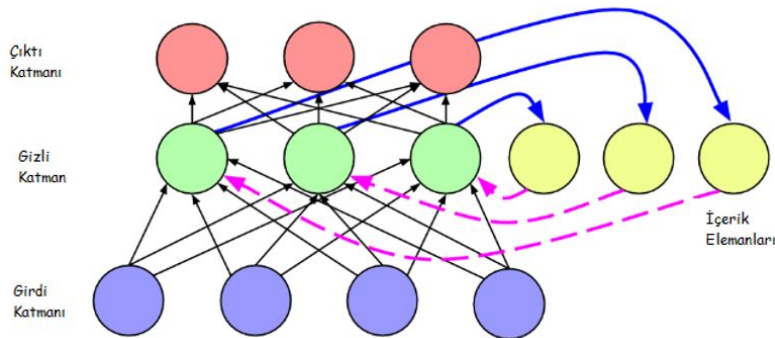
7.3) Geçitli Tekrarlayan Birim

Genel olarak Uzun Kısa Vadeli Hafızalara benzemektedirler. Geçitli Tekrarlayan Birim ve Uzun Kısa Vadeli Hafıza Birimleri, geleneksel RNN'lerin karşılaştığı kaybolan gradyan problemini ele alır, LSTM ise GRU'nun genelleştirilmiş halidir. Ancak Geçitli Tekrarlayan Birimlerin küçük veri setleri üzerinde daha iyi performans sergiledikleri görülmüştür.

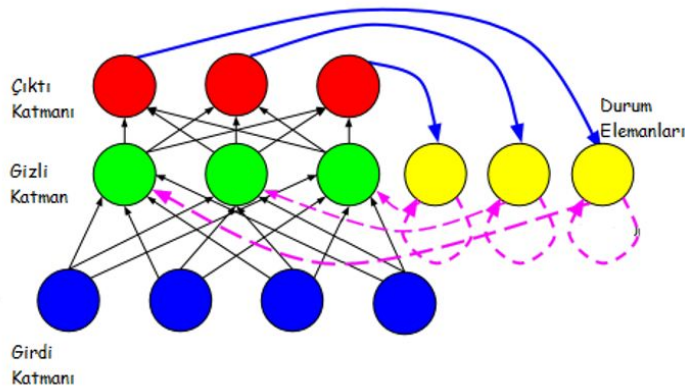
Karakterizasyon	Geçitli Tekrarlayan Birim (GRU)	Uzun Kısa Süreli Bellek (LSTM)
$\tilde{c}^{<t>}$	$\tanh(W_c[\Gamma_r \star a^{<t-1>}, x^{<t>}] + b_c)$	$\tanh(W_c[\Gamma_r \star a^{<t-1>}, x^{<t>}] + b_c)$
$c^{<t>}$	$\Gamma_u \star \tilde{c}^{<t>} + (1 - \Gamma_u) \star c^{<t-1>}$	$\Gamma_u \star \tilde{c}^{<t>} + \Gamma_f \star c^{<t-1>}$
$a^{<t>}$	$c^{<t>}$	$\Gamma_o \star c^{<t>}$
Bağımlılıklar		

7.4) Elman ve Jordan Ağları

Elman Ağı: Bu ağda girdi, çıktı ve gizli katmanının dışında İçerik Elemanları olarak adlandırılan elemanlar bulunmaktadır. Bu elemanlar gizli katmandaki değerleri bir sonraki iterasyona taşımakla yükümlüdür. İçerik Elemanları kendisi ile bağlantılı değildir ve gizli elemanlar ile arasındaki ağırlık katsayısı +1'dir.



Jordan Ağı: Çok katmanlı yapay sinir ağlarına benzeyen bu ağda girdi, çıktı ve gizli katmanın dışında Durum Elemanları olarak adlandırılan elemanlar bulunmaktadır. Bu elemanlar çıktı katmanındaki değerleri sonraki iterasyona girdi olarak taşımakla yükümlüdür. Ağa istenildiği kadar durum elemanı eklenebilir. Durum Elemanları kendisi ile bağlantılıdır ve çıktı elemanları ile arasındaki ağırlık katsayısı +1'dir. Bu durum ağın öğrenmesinin girdi katmanı ile gizli katman arasındaki ve gizli katmanla çıktı katmanı arasındaki bağlantılarda olmasını sağlar.



Elman

$$h_t = \sigma_h(W_h x_t + U_h h_{t-1} + b_h)$$

$$y_t = \sigma_y(W_y h_t + b_y)$$

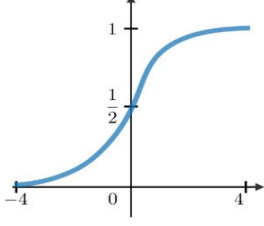
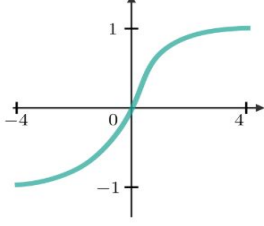
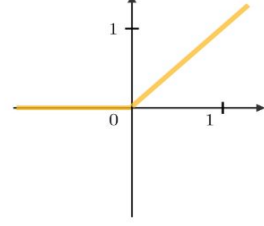
Jordan

$$h_t = \sigma_h(W_h x_t + U_h y_{t-1} + b_h)$$

$$y_t = \sigma_y(W_y h_t + b_y)$$

- x_t : Giriş vektörü
- h_t : Gizli katmanlı vektör
- y_t : Çıkış vektörü
- W , U ve b : Parametre matrisler ve vektör
- σ_h ve σ_y : Aktivasyon fonksiyonları

RNN modüllerinde kullanılan en yaygın aktivasyon fonksiyonları:

Sigmoid	Tanh	RELU
$g(z) = \frac{1}{1 + e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$
		

8-) Yinelenen Sinir Ağları Avantajlar ve Dezavantajlar

Avantajlar	Dezavantajları
<ul style="list-style-type: none">• Herhangi bir uzunluktaki girdilerin işlenmesi imkanı• Girdi büyüklüğüyle artmayan model boyutu• Geçmiş bilgileri dikkate alarak hesaplama• Zaman içinde paylaşılan ağırlıklar	<ul style="list-style-type: none">• Yavaş hesaplama• Uzun zaman önceki bilgiye erişme zorluğu• Mevcut durum için gelecekteki herhangi bir girdinin düşünülmemesi

9-) Yinelenen Sinir Ağları Problemleri

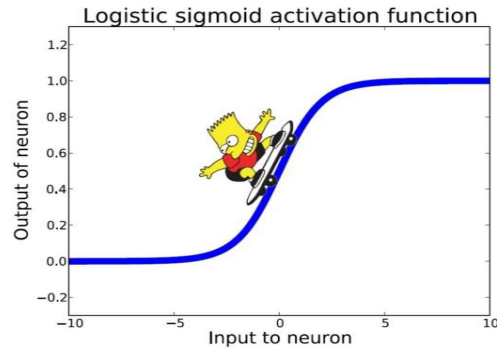
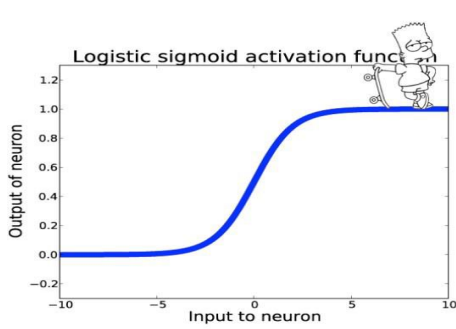
9.1) Kaybolan Gradyan Problemi

Nöral ağların ilkel formu olan basit algılayıcılar XOR kapısı gibi lineer olmayan problemler ile başa çıkamamaktaydı. Bunun sebebi aktivasyon fonksiyonu olarak kullanılan basamak fonksiyonunun türevi olmayan bir fonksiyon olmasıydı. Çoklu katman kullanımıyla birlikte kullanılan aktivasyon fonksiyonunun türevi alınabilen bir fonksiyon ile değişimini de gerektirmişti. Bu şekilde hataları geriye doğru yayabiliyor (geri yayılım algoritması backpropagation) ve öğrenimi sağlayabiliyorduk.

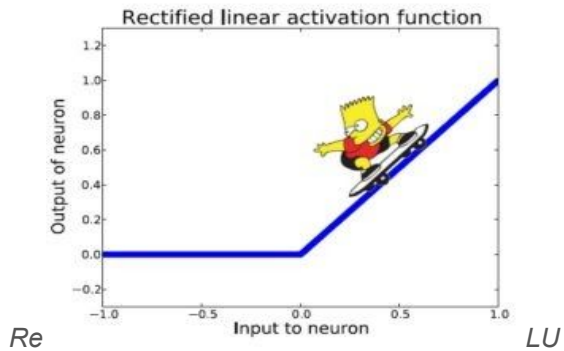
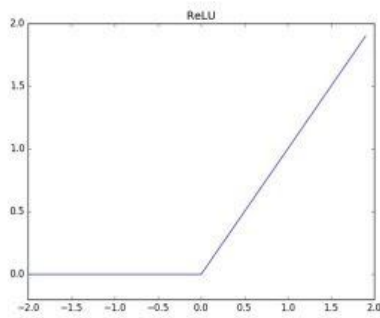
Ortaya atılması o günlere rastlayan lojistik sigmoid ve hiperbolik tanjant (tanh) fonksiyonları uzun yıllar popülerliğini sürdürecekti. Fakat bu fonksiyonlar da beraberinde büyük aksaklıklar getiriyordu. Sigmoid fonksiyonu -5 ve +5 girdileri arasında anlamlı sonuçlar üretmektedir. Fonksiyonun türevi bu aralıkta 0'dan farklıdır. Bu demek oluyor ki bu aralıktaki girdiler için türev alınabildiğinden hataları geriye doğru yayabilir ve öğrenimi sürdürebiliriz.

Ian Goodfellow, doktora tezinde bu anlamlı aralığı Bart Simpson'un kaykayı ile bu fonksiyon üzerinde hareket edebilmesi ile resmetmiştir. Yer çekimi Bart'ın [-5, +5] aralığında hareket etmesini sağlayacaktır. Öte yandan [-5, +5] aralığı dışında kalan bölgede yer çekimi Bart'ın hareket etmesine katkıda bulunmayacaktır.

Eğer aktivasyon fonksiyonumuzun çıktısı her zaman 0 çıktısı üretiyorsa, geri yayılım algoritması ile ağırlıkları güncelleyemeyiz. Tıpkı Bart'ın hareket edememesi gibi.

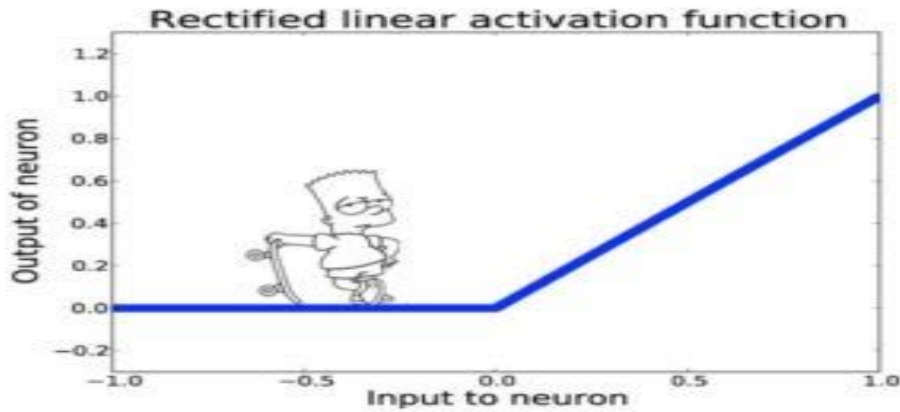


ReLU olarak adlandırılan çok basit bir aktivasyon fonksiyonunun ortaya atılması problemi ortadan kaldırmıştır. Bu fonksiyon pozitif girdiler için birim ya da özdeşlik fonksiyonu iken negatif girdiler için sıfır sonucunu üretmektedir.

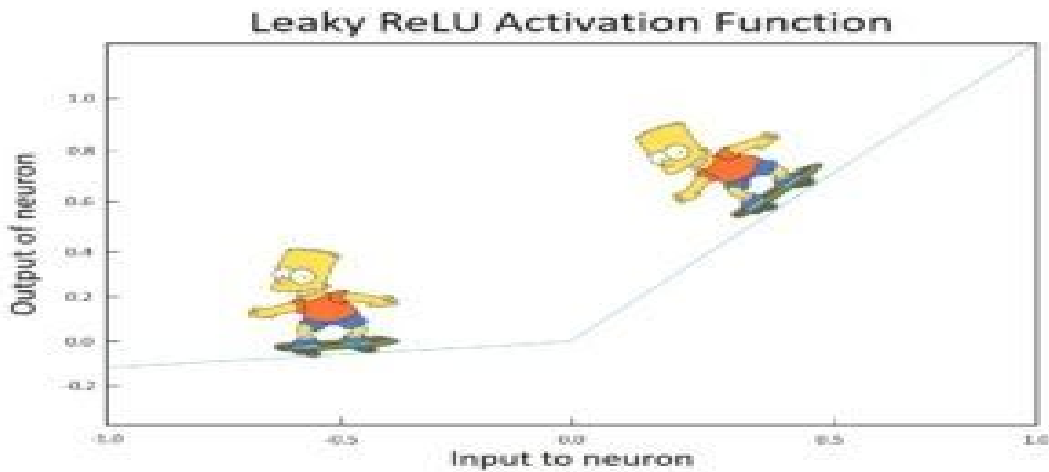


Derin ağların da çoğunlukla katmanlar boyunca büyük pozitif çıktılar ürettiğini göz önüne alırsak bu yeni fonksiyon ile büyük bir problem ortadan kalkacaktır.

Yine de bazı katmanlar büyük negatif çıktılar üretirse türev alınamayacak ve öğrenim gerçekleşemeyecektir.



Bu noktada sızıntılı ReLU (Leaky ReLU) aktivasyon fonksiyonu bunu çözebilmektedir. Pozitif bölgelerde daha hızlı olmak kaydı fonksiyonun (sıfır noktası haricinde) her noktasında türevi alınabilmekte ve sonuç elde edilebilmektedir.

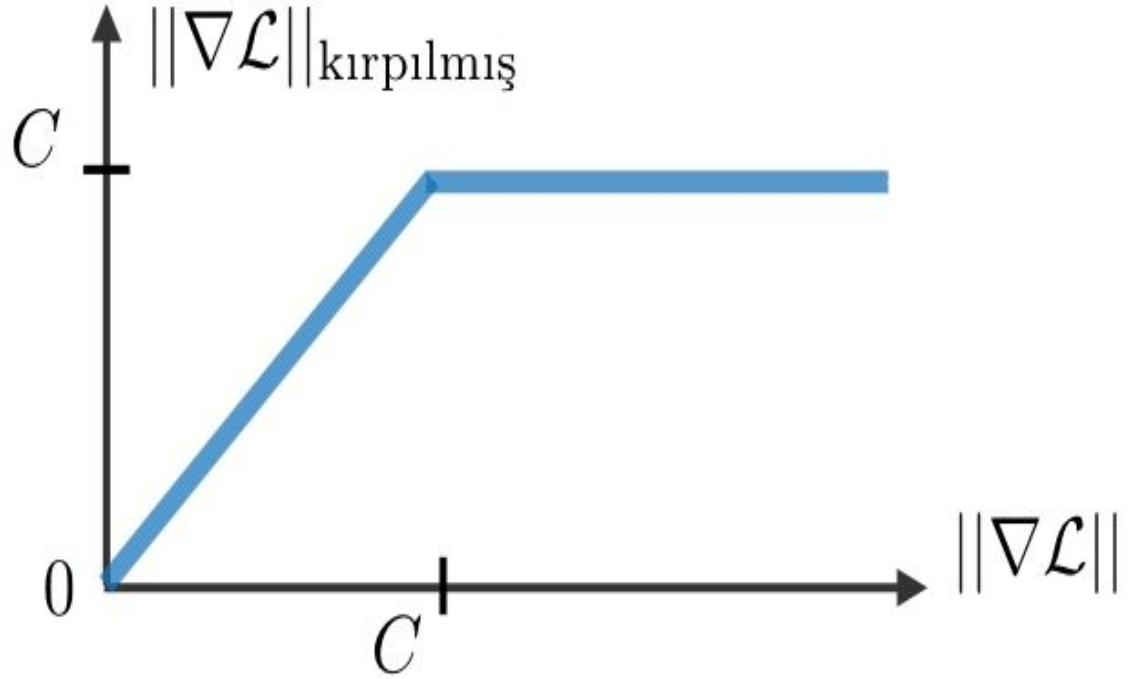


9.2) Patlayan Gradyan Problemi

Patlayan Gradyan Problemi, Makine öğreniminde Yapay Sinir Ağlarının gradyan bazlı ve geri yayılım algoritmalarıyla öğreniminde ortaya çıkan bir problemdir. Yapay Sinir Ağları girilen veriyi fonksiyonlar kullanarak anlayan ve özel bir çıkışa çeviren bir öğrenme algoritmasıdır. Bu tarz bir öğrenme algoritması insan beynindeki sinirleri taklit eder. Eğitim sırasında büyük gradyanlar biriktiğinde ise bu sinir ağına çok büyük güncellemeler gönderir ve patlayan gradyan probleminin ortaya çıkmasına sebep olur. Gradyanlar ağırlığını düzenlemede kullanılır, ama bu işlem en iyi güncellemeler küçük ve kontrollü yapıldığında çalışır.

Gradyanlar biriktiğinde ise kötü tahmin yapan veya hiçbir veri üretemeyen düzensiz bir ağ ortaya çıkması muhtemeldir, Bu problemi çözmek için gradyan kırpma ve ağırlık düzenleme gibi bazı metotlar vardır.

Gradyan kırpma — Geri yayılım işlemi sırasında bazen karşılaşılan patlayan gradyan sorunuyla başa çıkmak için kullanılan bir tekniktir. Gradyan için maksimum değeri sınırlayarak, bu durum pratikte kontrol edilir.



KAYNAKLAR:

- 1-<https://www.sciencedirect.com/topics/engineering/recurrent-neural-network>
- 2-<https://stanford.edu/~shervine/l/tr/teaching/cs-230/cheatsheet-recurrent-neural-networks#architecture>
- 3-https://tr.qwe.wiki/wiki/Recurrent_neural_network
- 4-<https://bilisim.io/2019/07/16/kaybolan-gradyan-problemine-bir-bakis/>
- 5-<https://deeptai.org/machine-learning-glossary-and-terms/exploding-gradient-problem>
- 6-https://www.researchgate.net/publication/322211817_DERIN_OGRENME_YONTEMLER_VE_UYGULAMALARI_HAKKINDA_BIR_INCELEME
- 7-<https://polen.itu.edu.tr/bitstream/11527/5861/1/7397.pdf>
- 8-<https://veribilimcisi.com/2017/09/26/tekrarlayan-sinir-aglari-recurrent-neural-network/>
- 9-<https://medium.com/@cansu.altunbas91/recurrent-neural-networks-rnn-yinelemeli-sinir-a%C4%9Flar%C4%B1-94feff6d7c3>
- 10-<https://medium.com/@ishakdolek/lstm-d2c281b92aac>
- 11-https://www.researchgate.net/publication/327666072_Derin_Ogrenme_ve_Turkiye%27deki_Uygulamalari
- 12-<https://devhunteryz.wordpress.com/2018/07/09/tekrarlayan-sinir-aglari-egitimi-bolum-1-rnnlere-giris/>
- 13-<https://www.derinogrenme.com/2017/03/04/yapay-sinir-aglari/>