



T.C

SAKARYA ÜNİVERSİTESİ

**BİLGİSAYAR VE BİLİŞİM BİLİMLERİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ**

BSM 310 – YAPAY ZEKA Yinelenen Sinir Ağları(RNN)

2/A

Grup üyeleri:

G171210014 ASLIHAN ÇETİNER

G181210350 ELYASE TUNAHAN SAĞLAM

G171210032 ESMA ÜLHÜSNA SİĞİRTMAÇ

G171210558 MUHAMMET ÖMER

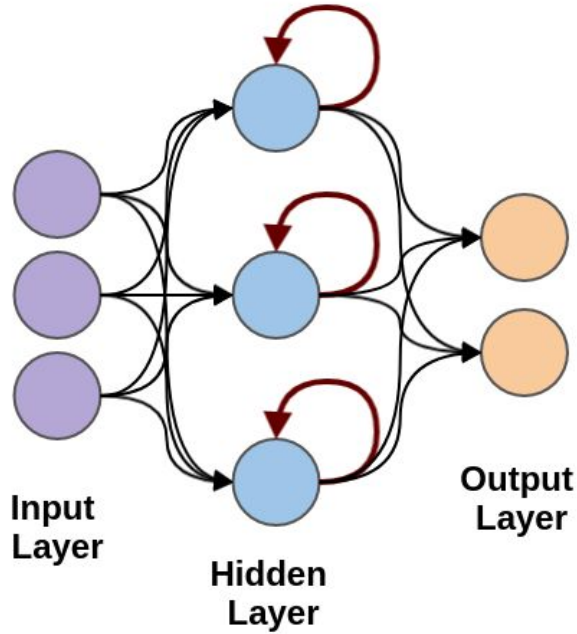
Sakarya

YİNELLENEN SİNİR AĞLARI (RNN)

Yinelenen sinir ağıları 1980'lerde geliştirilmiştir. Yapılan çalışmalar ile John Hopfield 1982 yılında Hopfield Ağlarını geliştirmiştir. Bu ağ ile birlikte bir sinirsel kompresör sisteminin zamanla nasıl gelişeceğini, 1000 adımdan sonra ki katmanların bile gerektirdiği durumları yinelenen sinir ağıları ile çözüme ulaştırılmış oldu.

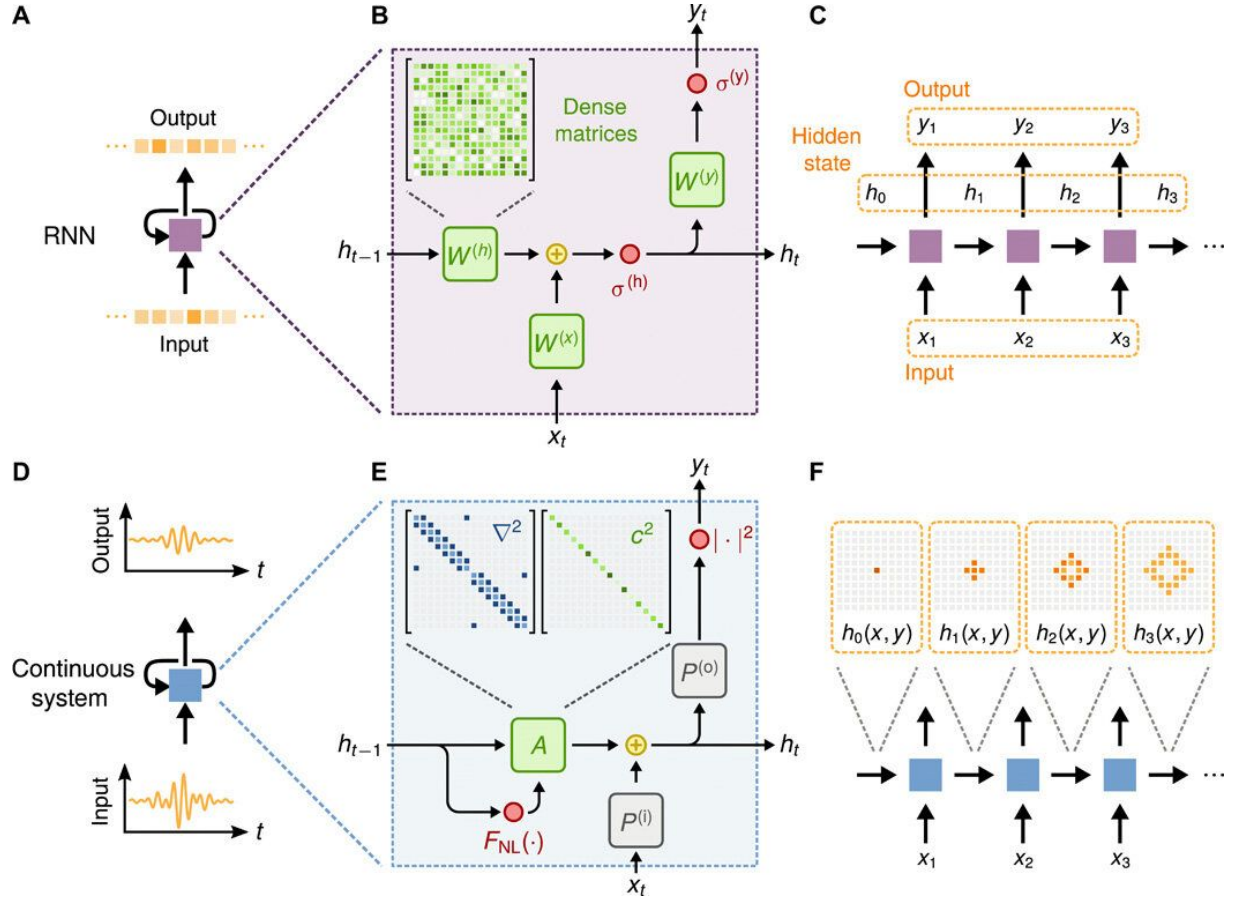
Yinelenen sinir ağıları, bir yapay sinir ağıdır. Düğümler arasında bağlantıları oluşturan bölgeler sinapslara sahiptirler ve sinir sistemiyle benzer şekilde çalışırlar. Yapay sinir ağlarından farklı olarak, yinelenen sinir ağıları giriş dizilerini işlemek için kendi belleklerini kullanabilirler.

Çıktıyı üretmek için tüm girdilerde veya gizli katmanlarda aynı görevi gerçekleştirildiğinden, her girdi için aynı parametreleri kullanır. Bu, diğer sinir ağlarının aksine parametrelerin karmaşıklığını azaltır. Bu bellek sayesinde el yazısı tanıma veya konuşma tanıma gibi hafıza gerektiren işlemlerde başarı sağlanmaktadır.



Temel yinelenen sinir ağıları, nöron benzeri birbirini takip eden tabakalar halinde, organize düğümler ve birbiri ile bağlantılı olan yönlendirilmiş bağlantılar sayesinde ardışık katmanlarda hareket ederler. Her bir düğümün, belli zamanda gerçekleşecek olan aktivasyon noktaları bulunur. Düğümler arası geçiş sinapslarla sağlanır ve geçişi sağlayan şey aktarılabilecek bilginin ağırlığıdır. Bu duruma göre sonra ki hücre gelen bilgiyi kabul eder veya sinapsal bölgede kalmasına karar verir. Bu hücreler giriş, çıkış ve gizli düğümlere sahiptir. Giriş düğümleri diğer hücrelerden gelen bilgileri uygunluğa göre yorumlayarak kabul etmekte ve işlenmesi için gizli düğümlere yollar. Gizli düğümler gelen bilgiyi, zamana bağlı aktivasyonuna göre etkileyerek çıkış düğümüne gönderir. Çıkış düğümünde bilgi son halini alır ve diğer işlemlerine devam edebilmesi için sinapsa bırakılır.

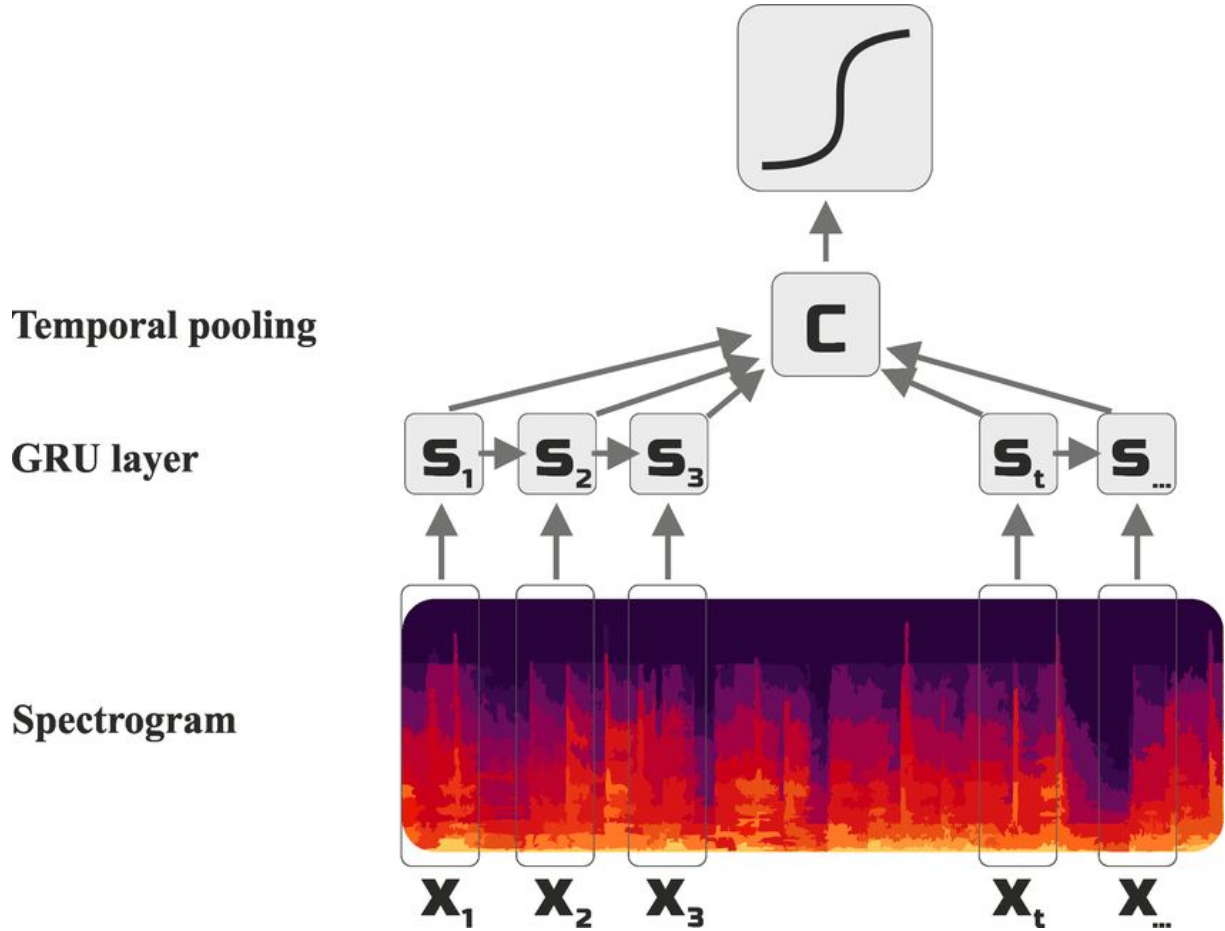
Her bir sekans ağı tarafında hesaplanan değere karşılık gelen aktivasyonlardan geçer ve tüm hedef sinyallerin sapma sonucuyla bir hata üretir. Üretilen bu hata, tüm birey dizilerin hatalarının toplamıdır ve eğitim setinde kullanılır. Her bir adımda hata oranının düşüşü beklenir.



Yinelenen sinir ağıları, birimler arasındaki bağlantıların yönlendirilmiş bir döngü oluşturduğu yapay sinir ağı sınıfıdır. Bu döngü ile, dinamik zamansal davranış sergilemesine olanak tanıyan bir ağı iç durumu oluşturulmuştur. İleri beslemeli sinir ağların aksine, Yinelenen sinir ağıları kendi giriş belleğini girdilerin rastgele dizilerini işlemek için kullanabilmektedirler. Tekrarlayan sinir ağındaki temel düşünce sıralı bilgileri kullanmaktır. RNN mimarisinin yinelenen olarak adlandırılmasının sebebi, bir dizinin her ögesi için aynı görevi önceki çıktılarına bağlı olarak yerine getirmesidir.

NEURO LINGUISTIC PROGRAMMING (NLP)

Duyu-Dil Programlama (DDP) anlamına gelen, kişilerin amaçlarına ulaşmaları için "nörolojik programlarını" keşfetmelerini ve en iyi şekilde kullanmalarını sağlamayı hedefleyen, tartışmalı bir psikolojik terapi anlayışıdır. Metodoloji modellemeye dayanır. NLP özel bir alanda başarılı sonuçlar alan kişilerin tecrübelerini biçimleme ve çözmeyi, NLP'nin anlaşılır olmasını ve ulaşmak isteyenleri de eğitmeyi sağlıyor. Modellemeden çıkan müdahale teknikleri kullanımı kolay, hızlı ve doğrulanabilir sonuçlar taşırlar.

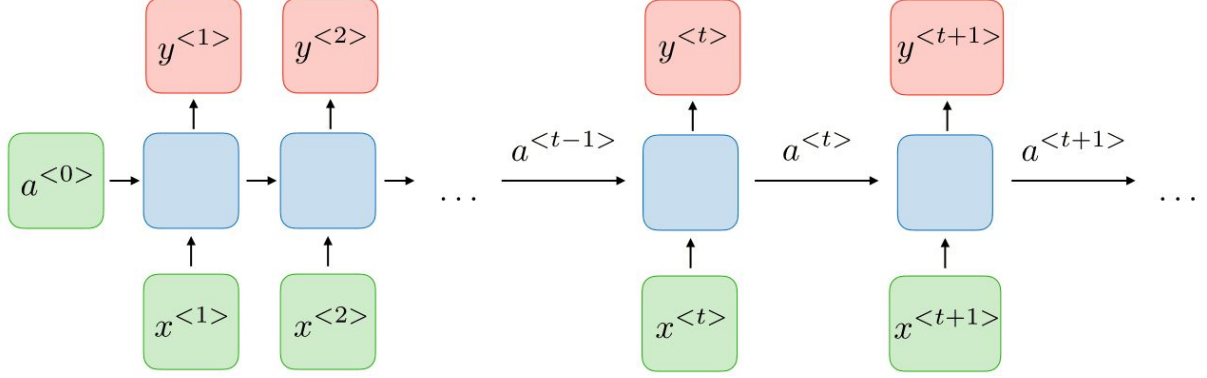


Yukarıda konuşurken ifade ettiğimiz duyguları geçici havuz yardımıyla sınıflandırmaya yarayan yinelenen sinir ağı modeli verilmiştir. Amaç öfke mutluluk gibi duyguların tahminidir.

GRU katmanında konuşmadaki pozitif negatif ve nötr ifadelerin anlamı kavranır. Transfer öğrenim yoluyla duyular sınıflandırılmış olur.

GELENEKSEL YİNELENEN SİNİR AĞ MİMARİSİ

Önce ki girdi, gizli aktivasyon ve çıktıların tekrardan girdi olarak kullanılmasını mümkün kılan mimari aşağıdaki gibidir.

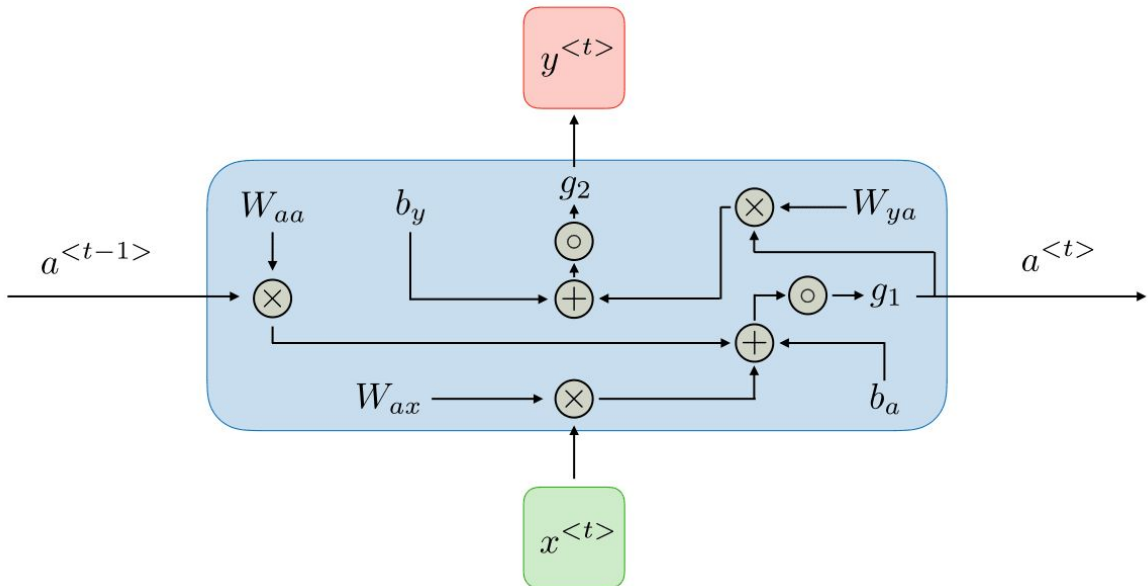


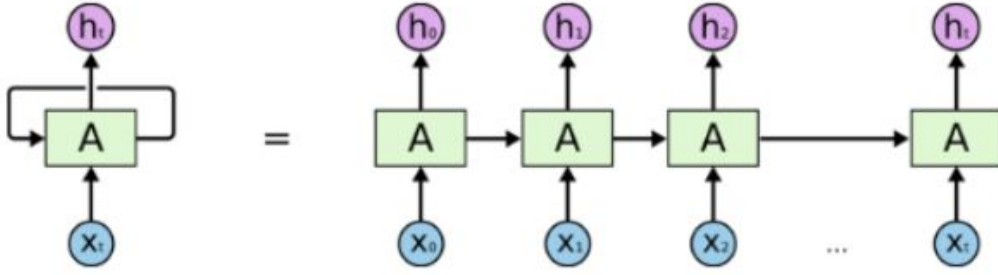
Her bir t zamanında $a^{<t>}$ aktivasyonu ve $y^{<t>}$ çıktısı ifade edilme aşağıdakiler gibidir;

- $a^{<t>} = g_1(W_{aa} a^{<t-1>} + W_{ax} x^{<t>} + b_a)$
- $y^{<t>} = g_2(W_{ya} a^{<t>} + b_y)$

W_{aa} , W_{ax} , W_{ya} , b_a ve b_y geçici olarak paylaşılan kat sayıdır ve g_1 ve g_2 aktivasyon fonksiyonlarıdır.

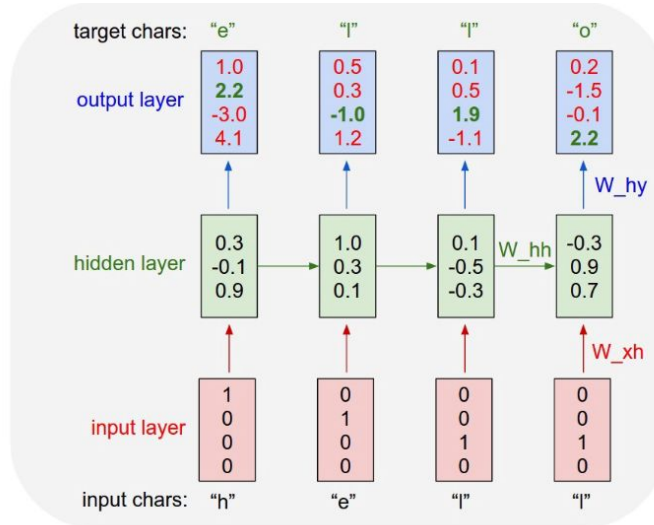
GENEL EĞİTİM TİPLEMESİ





İlk olarak, $X(0)$ 'ı giriş dizisinden alır ve daha sonra $X(1)$ ile birlikte bir sonraki adım için giriş olan $h(0)$ çıkışı verir. Bu nedenle, $h(0)$ ve $X(1)$ bir sonraki adımın girdisidir. Benzer şekilde, bir sonrakinden $h(1)$, bir sonraki adım için $X(2)$ ile girdi ve bu şekilde devam eder. Bu şekilde, eğitim sırasında içeriği hatırlamaya devam eder.

Şimdi bahsettiklerimizi aşağıdaki örnekle pekiştirelim diyagram bize “hell” girdisi ile beslenen yinelenen sinir ağlarının ileri doğru çalışan aktivasyon değerlerini göstermekte. Çıktı katmanı ise bize yinelenen sinir ağlarının sıradaki karakter için atadığı güven değerlerini göstermekte. Yeşil renkli sayıların yüksek, kırmızı renklilerin düşük olmasını istiyoruz.



Örneğin, ilk adımda yinelenen sinir ağı “h” karakterini aldığı anda sıradaki karakterin “h” olmasına 1.0, “e” olmasına “2.2”, “l” olmasına -3.0, “o” olmasına 4.1 güven değeri verdiğini görüyoruz. Eğitim verimiz “hello”daki sıradaki karakterin “e” olmasından dolayı, “e” için olan güven değerinin (yeşil) yüksek olmasını, diğerlerinin ise (kırmızı) düşük olmasını isteriz.

Neden Yapay Sinir Ağları?

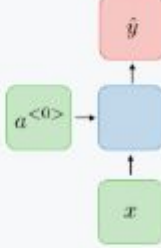
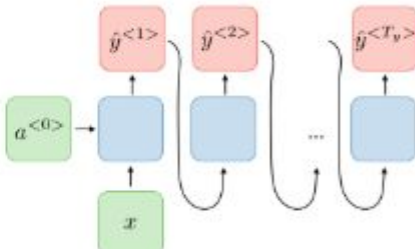
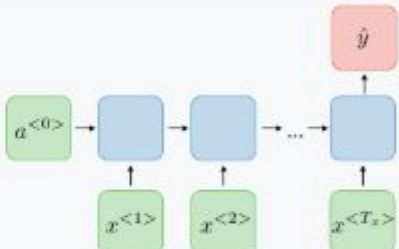
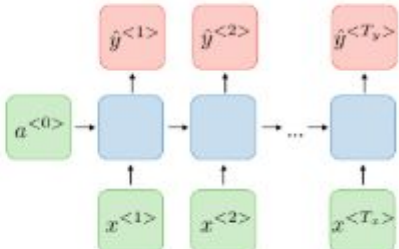
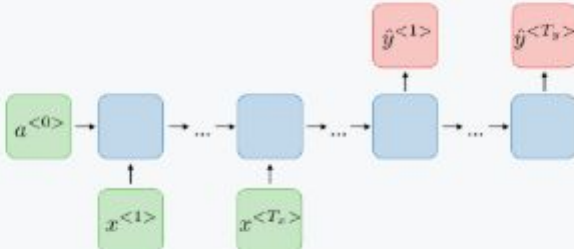
Hangi durumlarda Yapay Sinir Ağlarını kullanmalıyız? Hangi durumlarda kullanmamalıyız?

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

Zaman, projemizin önemli bir kriteri değil, geçmiş bilgileri sadece işlemek için kullanıyor isek yapay sinir ağı projemiz için uygun bir metot olabilir. Sabit model boyutu, girdilerinin uzunluğunun etki etmemesi, tüm bilgilerin dikkate alınması ve işin ağırlığının, ağın tamamına dağılması yinelenen sinir ağlarında fark yaratacak unsurlardır.

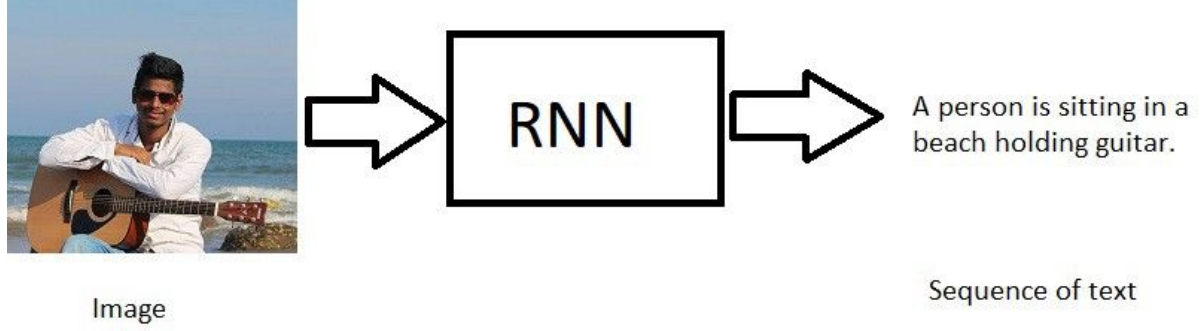
YİNELENEN SİNİR AĞLARI UYGULAMALARI

Yinelenen sinir ağı modelleri çoğunlukla doğal dil işleme ve konuşma tanıma alanlarında kullanılır. Farklı uygulamalar aşağıdaki tabloda özetlenmiştir:

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

Bire Çok İlişki

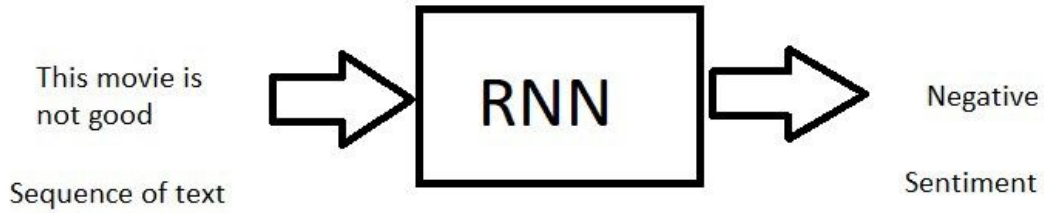
Yinelenen sinir ağıları, girdi olarak verilen bir resimden, çıktı olarak resmin eşleniği olan bir kelime dizisi üretebilir.



Yukarıda verilen örnekte tek bir fotoğraf girdi olarak verilirken, çıktı olarak “Sahilde elinde gitar tutan bir çocuk” kelime dizisi verilmiştir.

Çoka Bir İlişki

Yinelenen sinir ağıları girdi olarak aldığı kelime dizisini, tek bir çıktı olarak üretebilir.



Yukarıda verilen örnekte “bu film güzel değil” cümlesi verilmiştir ve çıktı ile bizde uyandırdığı duygu negatiftir.

Çoka Çok İlişkisi

Yinelenen sinir ağları sözcük dizisini girdi olarak alır ve sözcük dizisini çıktı olarak oluşturur.

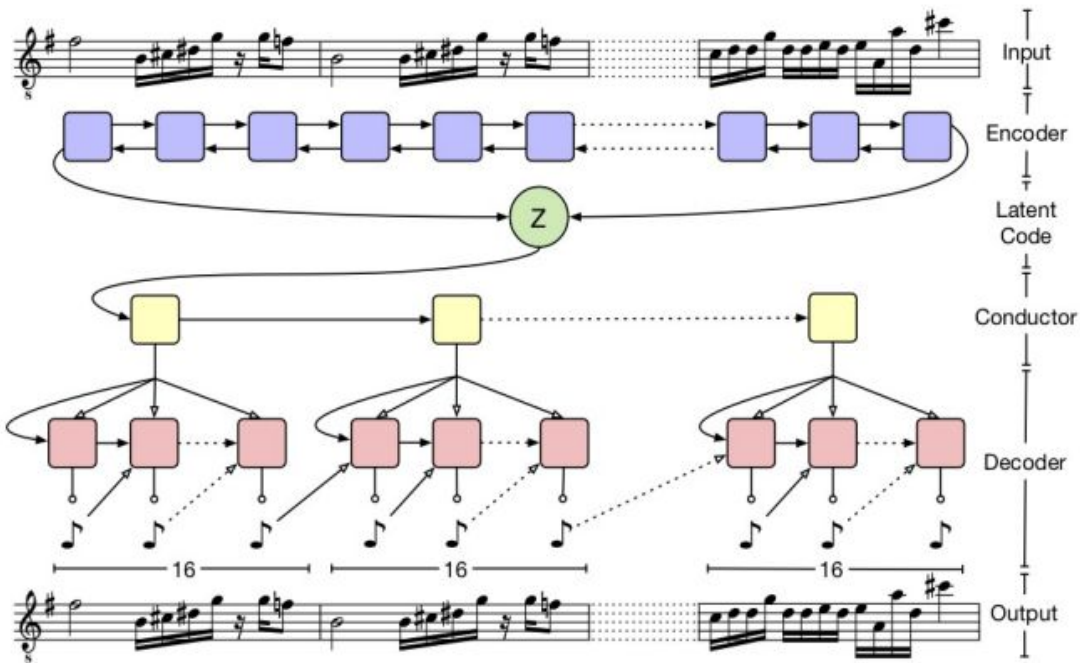


Yukarıda dil çevirisi örneğinden bahsedilmiştir. Hinduca verilen girdi “ Main tenu pyar karda haan ” çıktı olarak İngilizce “I love you” vermiştir.

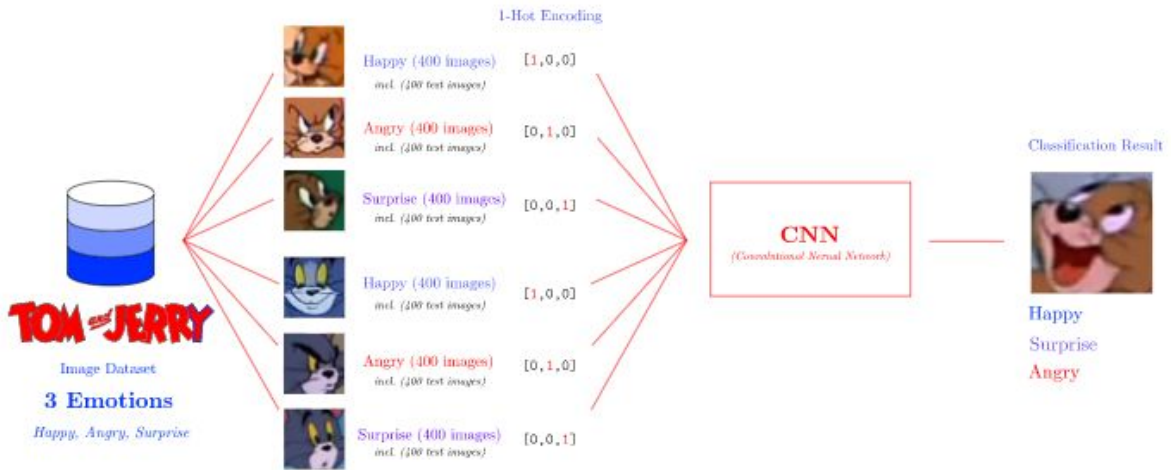
Yinelenen Sinir Ağları Uygulamalarına Örnekler

Yukarıda da bahsettiğimiz gibi yinelenen sinir ağları modelleri doğal dil işleme ve konuşma tanıma alanlarında sıklıkla kullanılır. Bu konuda çalışma yapan bilim insanlarından Alex Graves'in çalışmasında, ses verilerinin fonetik sunumuna gerek kalmadan doğrudan metine çeviren bir yinelenen sinir ağları tabanlı konuşma tanıma sistemi sunulmuştur . Başka bir çalışmada, yinelenen sinir ağları 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

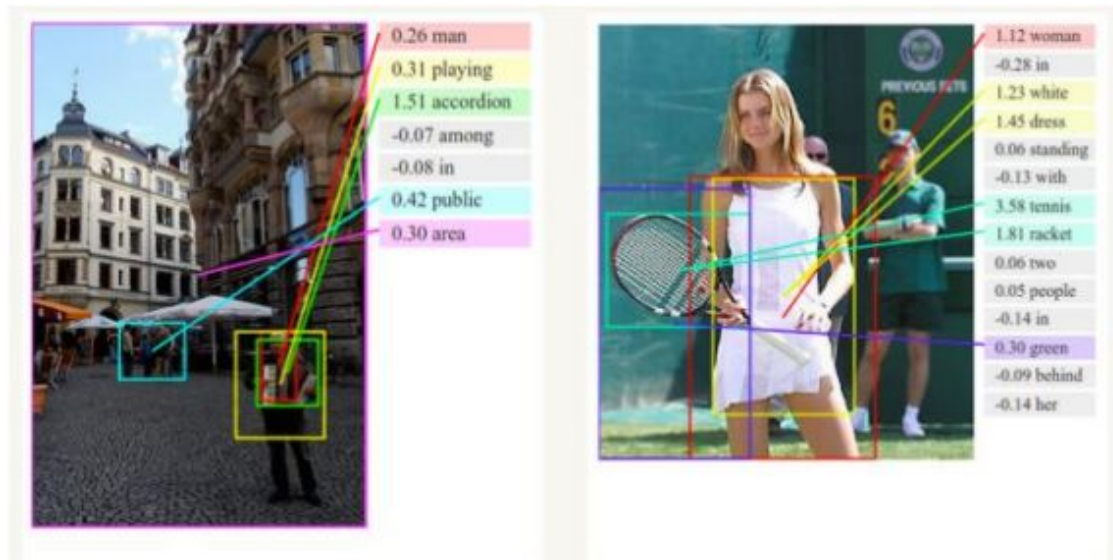
Müzik Üretimi



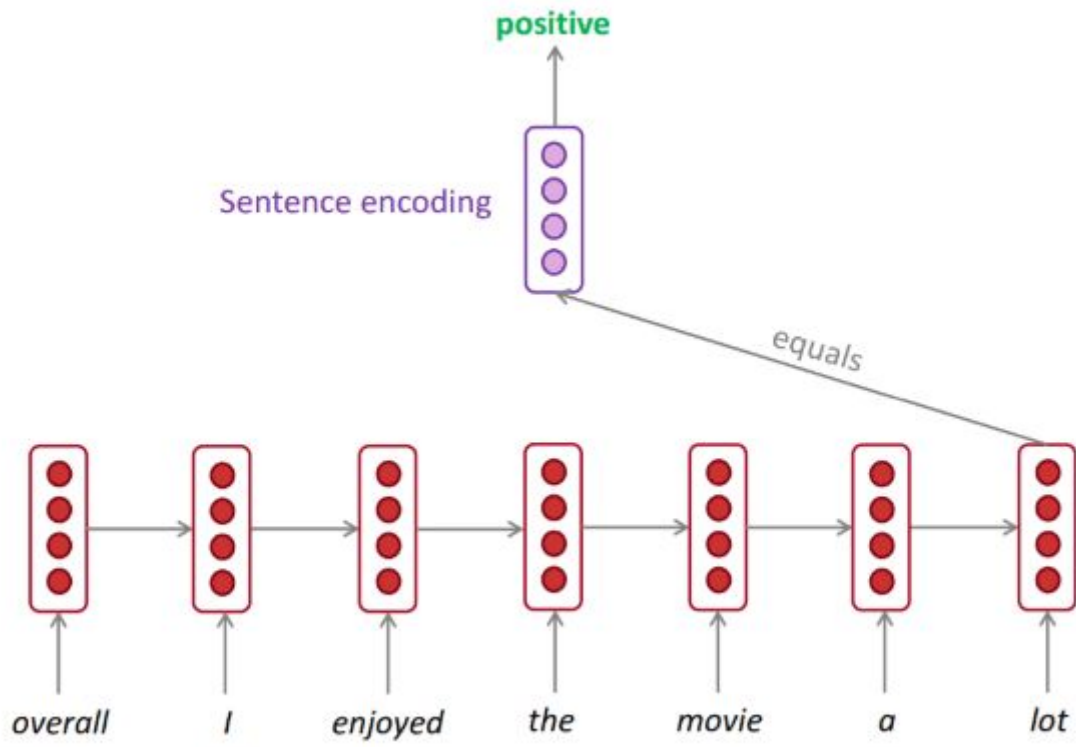
Duygu Sınıflandırma



İsim Varlık Tanıma



Makine Çevirisi

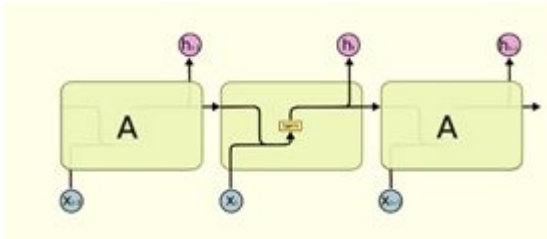


LONG-SHORT TERM MEMORY (LSTM)

Genellikle “LSTM’ler” olarak adlandırılan Long Short Term Memory ağırları, uzun süreli bağımlılıkları öğrenebilen özel bir YSA türüdür. Hochreiter & Schmidhuber (1997) tarafından tanıtıldılar ve sonraki çalışmalarda birçok kişi tarafından rafine edildi ve popülerleştirildi. Çok çeşitli problemler üzerinde çok iyi çalışıyorlar ve şimdi yaygın olarak kullanılıyorlar.

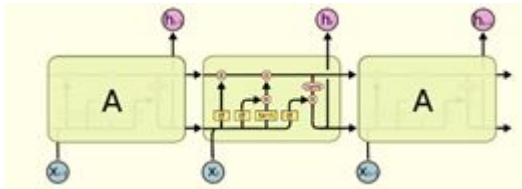
LSTM’ler, zaman ve katmanlar boyunca geri çoğaltılabilen hatanın korunmasına yardımcı olur. Daha sabit bir hatayı koruyarak, tekrarlayan ağların birçok zaman adımında (1000’den fazla) öğrenmeye devam etmesine izin verir, böylece nedenleri ve etkileri uzaktan bağlamak için bir kanal açar. Bu, makine öğrenimi ve yapay zeka için temel zorluklardan biridir, çünkü algoritmalar genellikle yaşamın kendisi gibi ödül sinyallerinin seyrek ve gecikmeli olduğu ortamlarla karşı karşıya kalır.

LSTM’ler uzun vadeli bağımlılık probleminden kaçınmak için açıkça tasarlanmıştır. Bilgiyi uzun süre hatırlamak pratikte varsayılan davranışlarıdır, öğrenmek için uğraştıkları bir şey değildir. Tüm tekrarlayan sinir ağları, sinir ağının tekrarlayan modülleri zinciri şeklindedir. Standart YSA’larda, bu tekrarlanan modül, tek tanh tabakası gibi çok basit bir yapıya sahip olacaktır.



Standart bir YSA’da yinelenen modül tek bir katman içerir.

LSTM’ler de bu zincir benzeri yapıya sahiptir, ancak tekrarlayan modül farklı bir yapıya sahiptir. Tek bir sinir ağı katmanına sahip olmak yerine, çok özel bir şekilde etkileşime giren dört tane vardır.



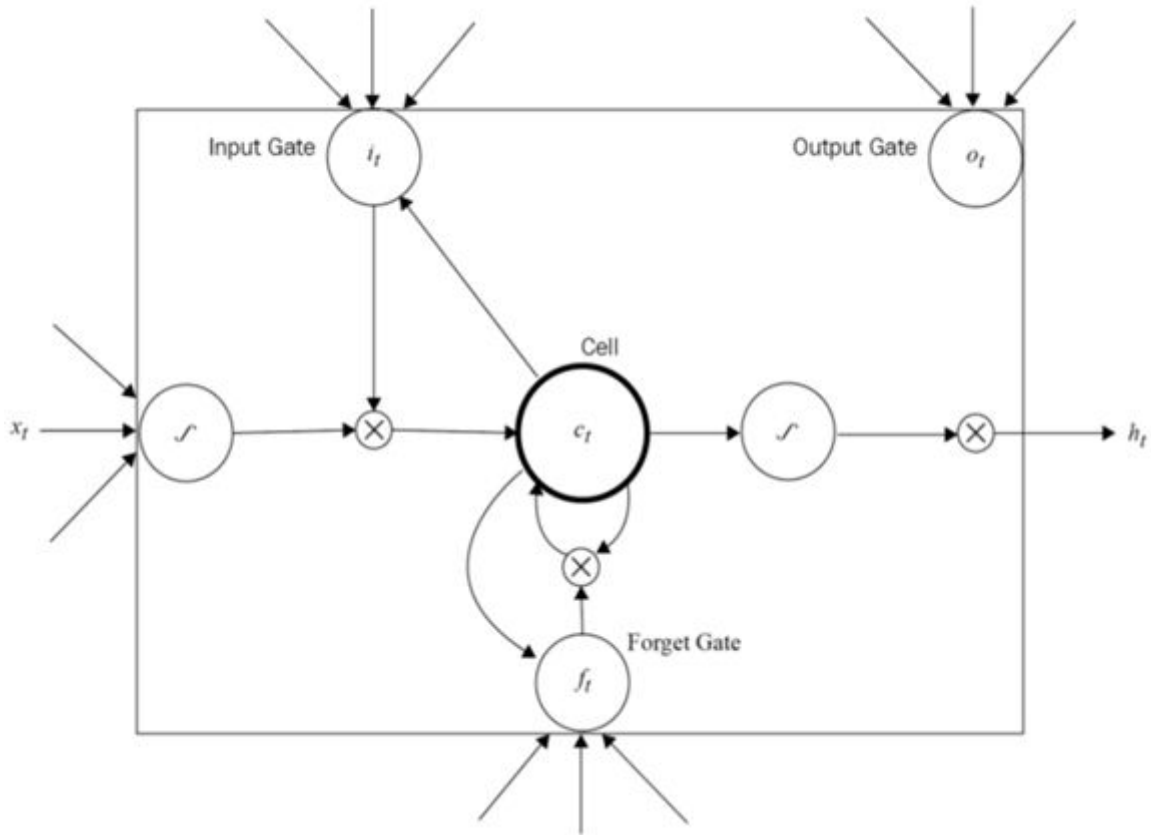
LSTM’deki yinelenen modül, etkileşen dört katman içerir.

Aşağıdaki diyagramda, her satır bir düğümün çıktısından başkalarının girişlerine kadar tüm vektörü taşır. Sarı kutular sinir ağı katmanları öğrenilirken, pembe daireler vektör eklenmesi gibi noktasal işlemleri temsil eder. Birleştirilen satırlar birleştirme hattını, çizgi çatallaştırma ise kopyalanan içeriği ve kopyaları farklı yere gideceklerini belirtir.



LSTM NASIL ÇALIŞIR?

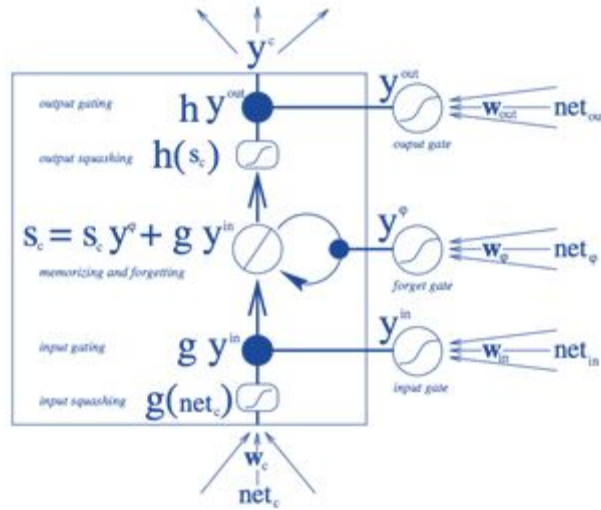
LSTM'ler, kapılı bir hücrede tekrarlayan ağı normal akışı dışında bilgi içerir. Bilgi, bir bilgisayarın belleğindeki veriler gibi bir hücrede saklanabilir, yazılabilir veya bir hücreden okunabilir. Hücre, neyin saklanacağı ve ne zaman açılıp kapanacakları ile okuma, yazma ve silme işlemlerine izin verileceğine karar verir. Bununla birlikte, bilgisayarlardaki dijital depolamanın aksine, bu kapılar analogdur ve hepsi 0-1 aralığında olan sigmoidlerle eleman bazında çarpma ile uygulanır. Analog, dijitalden farklı olma avantajına sahiptir ve bu nedenle geri çoğaltma için uygundur.



Bu

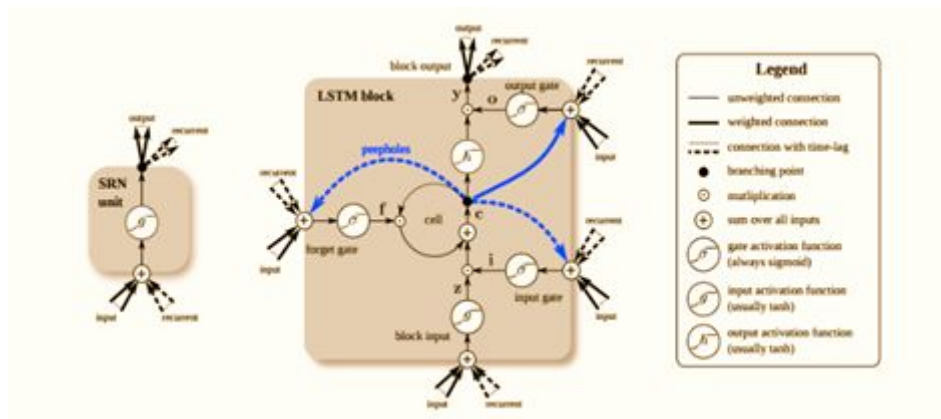
kapılar aldıkları sinyaller üzerinde hareket ederler ve sinir ağının düğümlerine benzer şekilde, kendi ağırlık kümeleriyle filtreledikleri güç ve içe aktarma temelinde bilgileri engeller veya iletirler. Bu ağırlıklar, girdi ve gizli durumları modüle eden ağırlıklar gibi, tekrarlanan ağların öğrenme süreci aracılığıyla ayarlanır. Yani, hücreler, tahmin yapma, geri yayılma hatası ve gradient iniş yoluyla ağırlıkların ayarlanması gibi yinelemeli süreç yoluyla verinin ne zaman girilmesine, bırakılmasına veya silinmesine izin verileceğini öğrenir.

Aşağıdaki şema, verinin bir bellek hücresinden nasıl aktığını ve kapıları tarafından nasıl kontrol edildiğini gösterir.



- Alttan başlayarak, üçlü oklar bilginin hücreye birden fazla noktada aktığını gösterir. Mevcut giriş ve geçmiş hücre durumunun kombinasyonu sadece hücrenin kendisine değil, aynı zamanda girişin nasıl işleneceğine karar verecek olan üç kapısının her birine beslenir.
- Siyah noktalar, sırasıyla yeni girdinin içeri girip girmeyeceğini, mevcut hücre durumunu silip silmeyeceğini ve / veya bu durumun, ağın çıktısını günümüzde etkilemesine izin verip vermediğini belirleyen kapılardır. s_c , bellek hücresinin geçerli durumudur ve $g_{y_{in}}$, hücrenin geçerli girdisidir. Her kapının açık veya kapalı olabileceğini ve her adımda açık ve kapalı durumlarını yeniden birleştireceklerini unutmayın. Hücre durumunu unutulabilir ya da unutamaz; yazılı olsun ya da olmasın; ve her zaman adımında okunsun veya okunmasın, ve bu akışlar burada temsil edilir.
- Büyük kalın harfler bize her işlemin sonucunu verir.

Basit bir yinelenen ağı (solda) bir LSTM hücresiyle (sağda) karşılaştırarak iyi bir ölçüm için başka bir diyagram:



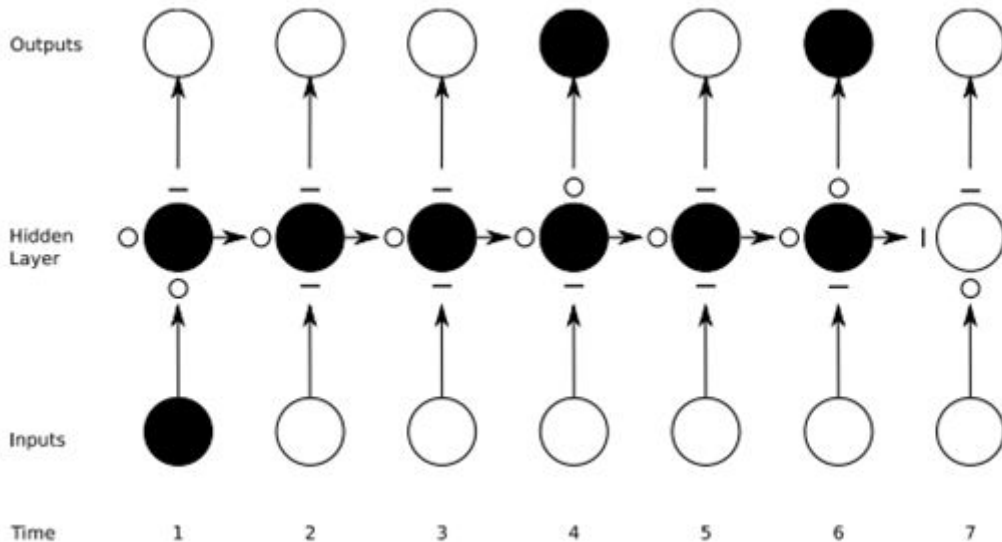
LSTM'lerin bellek hücrelerinin, girdinin dönüştürülmesinde toplama ve çarpma işleminde farklı roller verdiğini belirtmek önemlidir. Her iki diyagramdaki merkezi artı işareti esasen LSTM'lerin sırrıdır. Görüldüğü gibi basit olan bu temel değişiklik, derinlemesine geri yayılması gerektiğinde sabit bir hatayı korumalarına yardımcı olur. Geçerli durumunu yeni girdiyle çarparak sonraki hücre durumunu belirlemek yerine, ikisini ekler ve bu tam anlamıyla fark yaratır.

Farklı ağırlık setleri giriş, çıkış ve unutma girişlerini filtreler. Unutma kapısı doğrusal bir kimlik fonksiyonu olarak temsil edilir, çünkü kapı açıksa, bellek hücresinin mevcut durumu, bir kez daha ileri doğru ilerlemek için basitçe bir ile çarpılır.

Dahası, her bir LSTM hücresinin unutulma geçidine 1'lik bir sapma da dahil olmak üzere basit hackler konusunda da performansı artırdığı gösterilmiştir

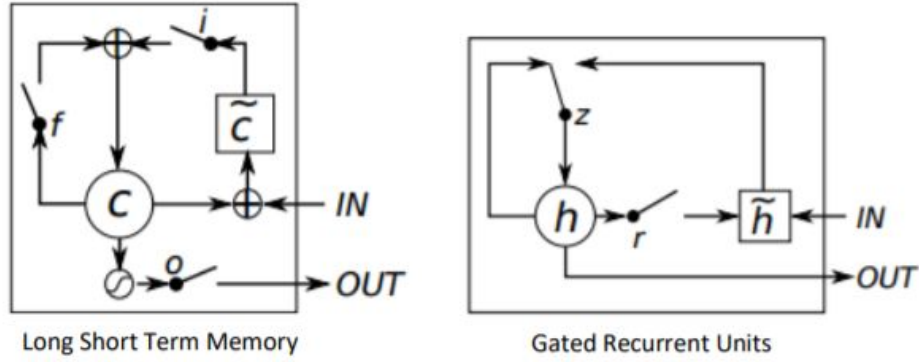
LSTM'lerin amacı uzak olayları nihai bir çıktıya bağlamak olduğunda neden unutulmuş bir kapıya sahip olduğunu merak edebilirsiniz. Bazen unutmak iyidir. Örneğin, bir metin topluluğunu analiz ediyorsanız ve bir belgenin sonuna geliyorsanız, bir sonraki belgenin onunla herhangi bir ilişkisi olduğuna inanmanız için hiçbir nedeniniz olmayabilir ve bu nedenle bellek hücresi, net, bir sonraki belgenin ilk ögesini alır.

Aşağıdaki şemada, kapalı kapıları temsil eden düz çizgiler ve açık kapıları temsil eden boş daireler bulunan iş kapılarını görebilirsiniz. Gizli katmandan aşağıya doğru yatay olarak uzanan çizgiler ve daireler unut kapılarıdır.



GATED RECURRENT UNITS (GRUs)

Bir geçitli yinelenen birim (GRU) temelde bir çıkış geçidi olmayan bir LSTM'dir, bu nedenle içeriği her zaman adımında bellek hücresinden daha büyük ağıta tam olarak yazar.



LSTM ve GRU arasında birçok benzerlik vardır. Bununla birlikte, hatırlanmaya değer bazı önemli farklılıklar vardır:

- Bir GRU'nun iki kapısı varken, bir LSTM'nin üç kapısı vardır.
- GRU'lar, açıkta kalan gizli durumdan farklı bir dahili belleğe sahip değildir. LSTM'lerde bulunan çıkış kapısı yok.
- GRU'da çıktı hesaplanırken ikinci bir doğrusallık uygulanmaz.

Kavram olarak GRU, LSTM'den biraz daha yenidir. Genellikle daha verimlidir, modelleri LSTM'den daha hızlı eğitir. Ayrıca kullanımı daha kolaydır. Bir modelde yapmanız gereken değişiklikler oldukça kolay bir şekilde yapılabilir. Ancak LSTM, daha uzun süreli belleğin gerekli olduğu GRU'dan daha iyi performans göstermelidir. Sonuçta, performansı karşılaştırmak kullandığınız veri kümesine bağlı olacaktır.

ZAMANA GÖRE GERİ YAYILIM (BPTT)

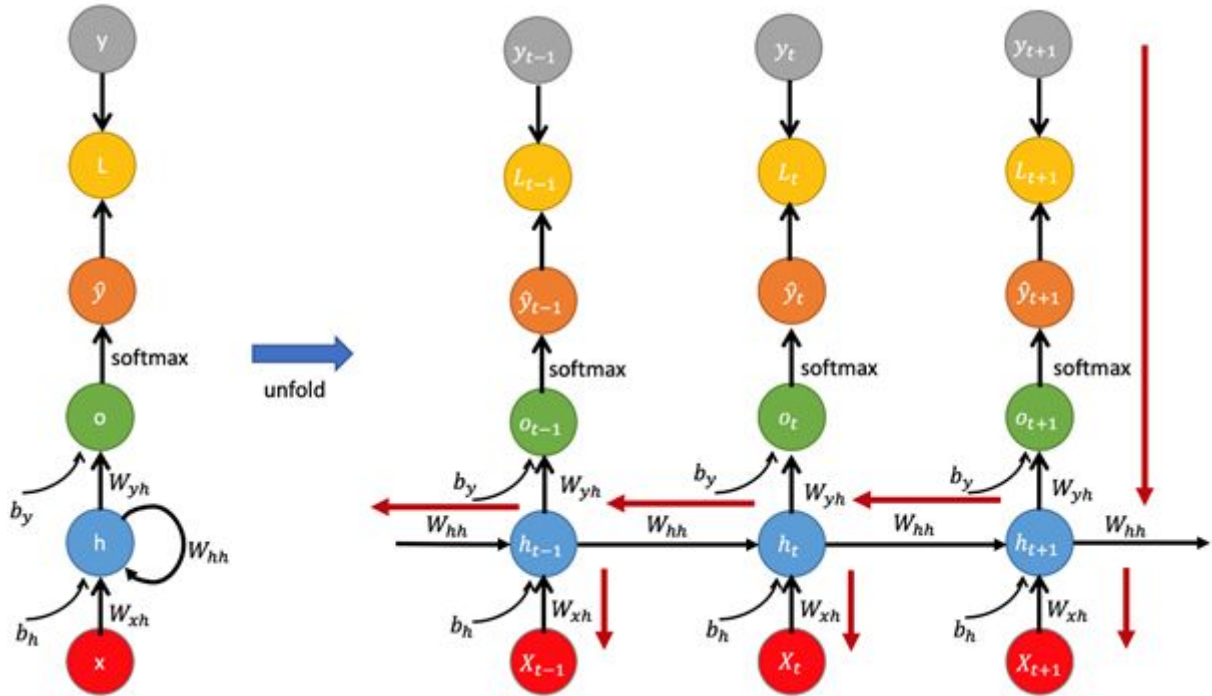
Zaman İçinde Geri Yayılım veya BPTT, LSTM'ler gibi tekrarlayan sinir ağlarındaki ağırlıkları güncellemek için kullanılan eğitim algoritmasıdır.

BPTT'ye giriş yapmadan geri yayılım eğitim algoritmasına kısaca değinebiliriz çünkü ikisi birbirleriyle bağlantılı olan kavramlardır.

Geri yayılım eğitim algoritmasının amacı, karşılık gelen girişlere yanıt olarak ağ çıkışlarının hatalarını en aza indirmek için bir sinir ağına ağırlıklarını değiştirmektir.

Ağın, yapılan belirli hatalarla ilgili olarak düzeltilmesini sağlayan denetimli bir öğrenme algoritmasıdır.

Geri eğitim algoritması, sabit boyutlu giriş-çıkış çiftlerinde ileri beslemeli sinir ağlarını eğitmek için uygundur, ancak geçici olarak sipariş edilebilecek dizi verileri hakkında ne söylenebilir?



İşte tam da burada Zaman İçinde Geri Yayılım (BPTT) devreye girer.

En basit haliyle BPTT; Geri Yayılım Eğitim algoritmasının bir zaman serisi gibi dizi verilerine uygulanan tekrarlayan sinir ağına uygulanmasıdır.

Tekrarlayan bir sinir ağı, her zaman aralığında bir giriş gösterilir ve bir çıkışı tahmin eder.

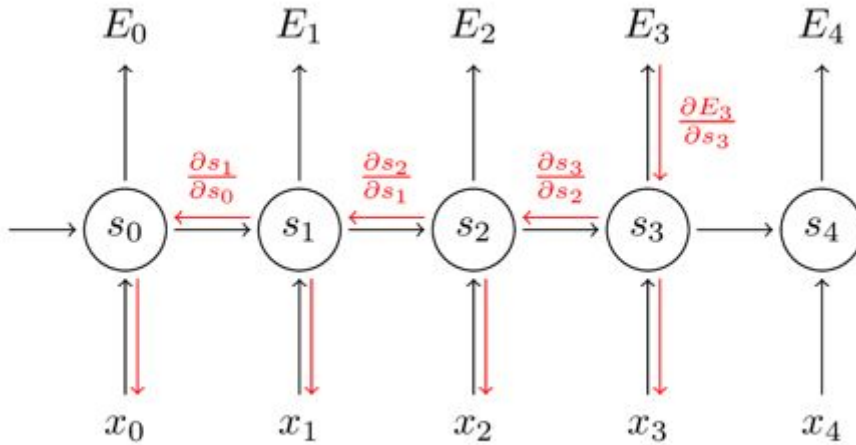
Kavramsal olarak, BPTT tüm girdi zaman aralıklarını açarak çalışır. Her zaman adımının bir giriş zaman aralığı, ağı bir kopyası ve bir çıkışı vardır. Daha sonra hatalar hesaplanır ve her zaman aralığı için birikir. Ağ geri alınır ve ağırlıklar güncellenir.

Algoritmayı şu şekilde özetleyebiliriz:

1. Ağa giriş ve çıkış çiftlerinin zaman aralıklarını gösterme.
2. Ağı açın, ardından her bir zaman aralığında hataları hesaplayın ve biriktirin.
3. Ağı toplayın ve ağırlıkları güncelleyin.
4. Tekrar et.

BPTT, zaman akışı sayısı arttıkça hesaplama açısından pahalı olabilir.

Giriş dizileri binlerce zaman diliminden oluşuyorsa, bu tek bir güncelleme ağırlığı güncellemesi için gerekli türev sayısı olacaktır. Bu, ağırlıkların yok olmasına veya patlamasına (sıfıra veya taşmaya) neden olabilir ve yavaş öğrenme ve model becerisini gürültülü hale getirebilir.



Slaytta da bulunan bu örneğimizi ayrıntılı bir şekilde ele alalım. Yukarıda sıralı 5 girdili bir recurrent ağ yapısı gösterilmektedir. E hatayı göstermektedir. Örneğin, E3 için backpropagation yaparken yaptığımız işlemde w ağırlığına göre türevi kullanılmaktadır. Bu türevi çözebilmek için zincir kuralı ile birkaç türevin çarpımını kullanırız.

$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial W}$$

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^3 \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W}$$

Bu çarpım yukarıda gösterildiği gibidir. Burada s3'ün açılımında s2'ye bir bağımlılık bulunur. Bunu çözebilmek için yine zincir kuralını kullanarak s3'ün s2'ye türevini de ekleyerek sonucu bulabiliriz. Böylece formüldeki gibi gradient zamana bağlı şekilde dağıtılmış olur. Parametreler ağıdaki tüm zaman aşamaları tarafından paylaşıldığı için, her çıkıştaki gradyan sadece geçerli zaman adımının hesaplarına değil, aynı zamanda önceki zaman adımlarına da bağlıdır.

Bahsedilmiş Bazı Kavramlar

Kaybolan/patlayan gradyan — Kaybolan ve patlayan gradyan fenomenlerine RNN'ler bağlamında sıklıkla rastlanır. Bunların olmasının nedeni, katman sayısına göre katlanarak azalan/artan olabilen çarpımsal gradyan nedeniyle uzun vadeli bağımlılıkları yakalamanın zor olmasıdır.

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.

Giriş kapıları çeşitleri — Kaybolan gradyan problemini çözmek için bazı RNN türlerinde belirli kapılar kullanılır ve genellikle iyi tanımlanmış bir amaca sahiptir.

Sorular

1)Basit Tekrarlayan Ağ (SRN), kim tarafından tasarlanmıştır?

- A) Sepp Hochreiter
- B) Juergen Schmidhuber
- C) Jeff Elman**
- D)David Rumelhalt

2) LSTM yapısında tekrar eden modül özel bir şekilde bağlı 4 katman bulunur. Hangileri bu modülde bulunan bilgilerin özelliğidir?

- I- Depolanabilirlik
- II- Hücreye Yazılabilme
- III- Okunabilirlik
- IV- Filtreleme

- A) Yalnız I
- B) I - II - III**
- C) I - II - III - IV.
- D) Yalnız IV

3) Hangisi RNN kullanılan alanlara girmez ?

- A) Dil Modelleme ve Metin Oluşturma
- B) Konuşma Tanıma
- C) Makine Çevirisi
- D) Veri İletimi**

4) Kaynak dilimizin İngilizce olduğu bir modelleme işleminde giriş dizisinin “Awes” olduğu durumda beklenen tahmini çıktı ne olacaktır?

- A) Awful
- B) Allies
- C) Awaydays
- D) Awesome**

KAYNAKÇA

- <http://colah.github.io>
- <https://deeplearning4j.org>
- <https://ayearofai.com>
- <http://www.wildml.com>
- <http://curiously.com>
- <https://medium.com/@mliuzzolino/hello-rnn-55a9237b7112>
- <http://www.wildml.com>
- <https://slidesgo.com/>
- <https://pathmind.com/wiki/lstm>