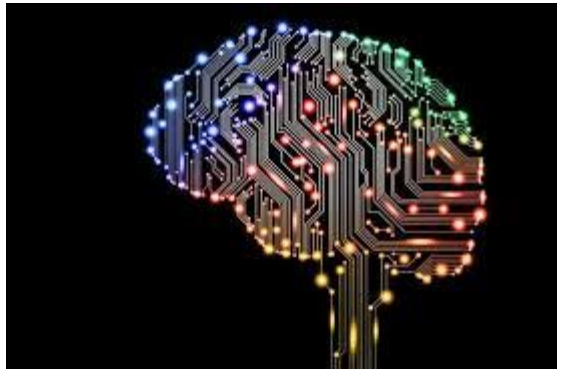
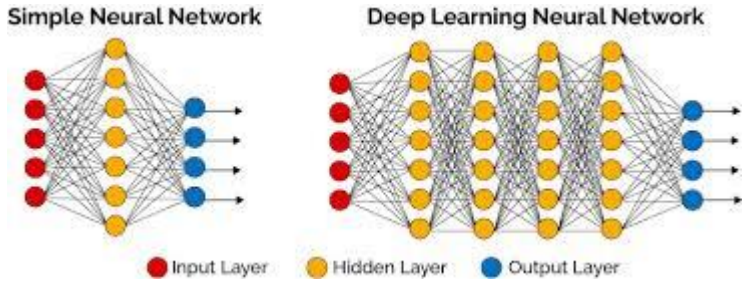
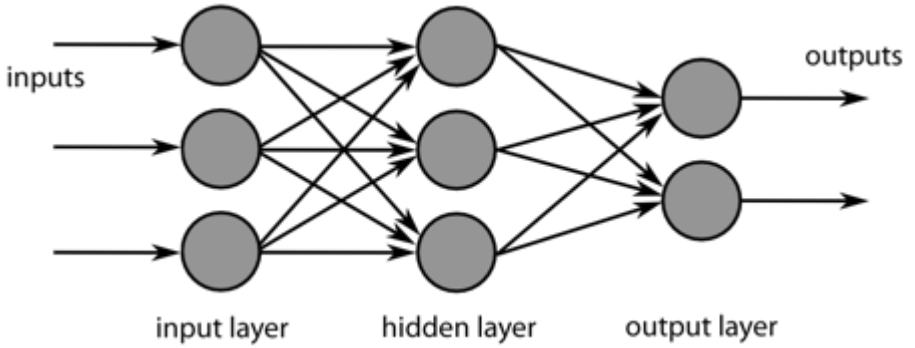




Derin Öğrenme Modelleri

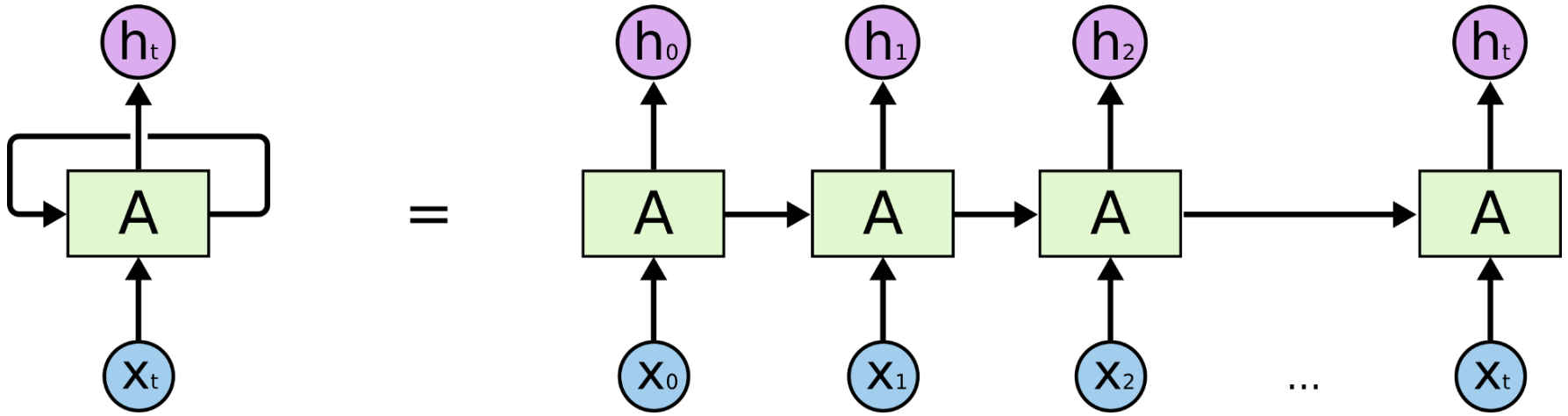
İleri Beslemeli Ağlar – Feed Forward Network

- Gelen bilgi sadece ileri doğru işlenir.
- Giriş verileri ağdan geçirilerek bir çıkış değeri elde edilir. Çıkış değerleri gerçek değerler ile karşılaştırılarak hata hesaplanır.
- Ağ üzerindeki ağırlık değerleri hataya bağlı olarak değiştirilir.
- En doğru çıktıyı verebilen bir model oluşturulur.
- FF bir ağın eğitiminde hatanın minimize edilmesi gerekir.
- Nöronlara giden ağırlıklar yenilenerek girdiye uygun çıktı veren bir yapı oluşturulur.
- Zamana veya sıraya bağlı değildir, sadece o an ki girdi verisi ile ilgilenir.

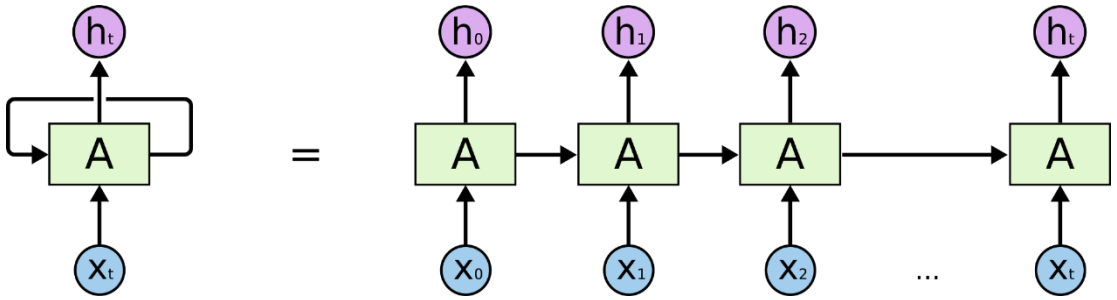


RNN (Recurent Neural Network-Yinelenen Sinir Ağları) Nedir?

- Bir sonraki adımı tahmin etmek için kullanılan bir Derin Öğrenme modelidir.
- En büyük farkları ***hatırlamalarıdır***.
- Diğer sinir ağlarında her girdi birbirinden bağımsız iken, RNN'lerde girdiler birbirleri ile ilişkilidir.
- RNN'ler bir sonraki adımı takip edebilmek için girdiler arasında ilişki kurarlar ve eğitilirken tüm bu ilişkileri hatırlarlar.
- Hatırlama nasıl oluyor ? Kurmuş oldukları ilişkilerin kalıcı olması için kendi içlerinde döngü benzeri bir yapı kullanırlar.



Kaynak:https://medium.com/@vidishajitani25/recurrent-neural-network-maths-69214e4d69e1



$$h_t = f(h_{t-1}, x_t)$$

h_t : Şu anki h değeri
h_{t-1}: Bir önceki h değeri
x_t : Şu anki girdi vektörü

Bir aktivasyon fonksiyonu ile kullanalım

$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

W : Ağırlık
h : Gizli katman
W_{hh} : Bir önceki gizli katmanın ağırlığı
W_{xh} : Şu anki gizli katmanın ağırlığı
tanh : Aktivasyon fonksiyonu

$$y_t = W_{hy}h_t$$

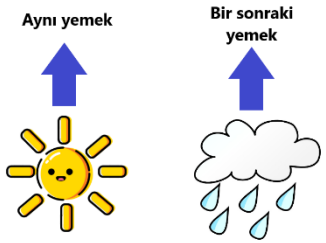
W_{hy} : Çıktı katmanının ağırlık değeri
y_t : Çıktı

- RNN, çıktılarını bir sonraki işleme girdi olarak verdikleri için FF yapılardan farklıdırlar. RNN'ler bir belleğe sahiptir diyebiliriz.
- Bir ağa, bellek eklemenin sebebi, belli bir düzende gelen girdi verisinin, çıktı için bir anlamı olmasıdır.
- Bu çeşit girdi verileri için FF ağlar yeterli olmaz.
- Yazı, konuşma ve zamana bağlı çeşitli sensörlerden belli bir sıra ile gelen verilerin yapısını anlamada kullanılırlar.

RNN Bir örnek ile açıklayalım

❖ Anneniz sizin için yemek yapıyor ama kuralları var. Yemeklerin yapılış sırası ***elmalı turta***, ***hamburger*** ve ***tavuk***.

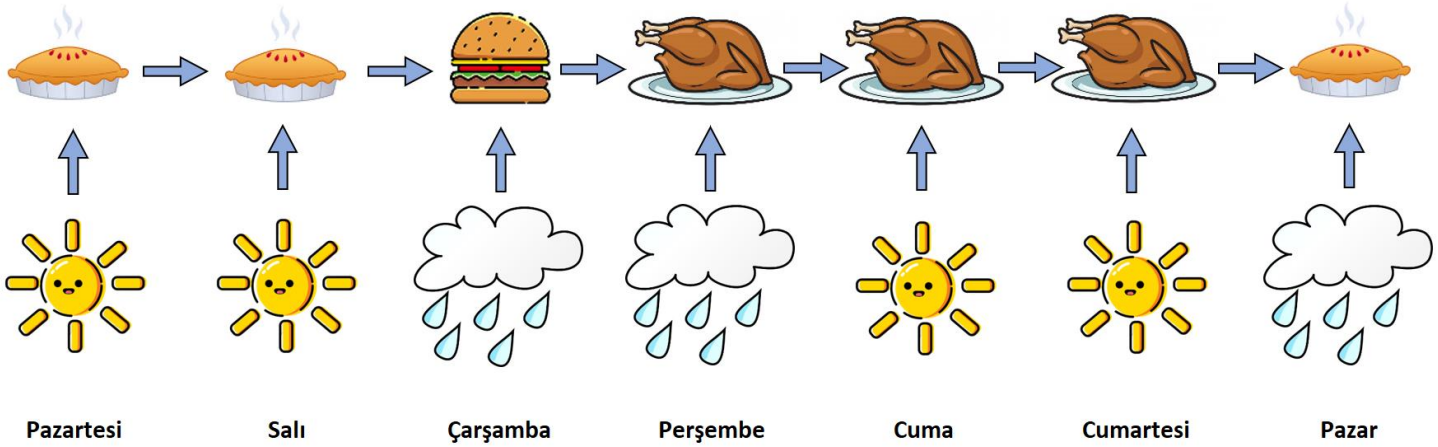
Pazartesi Salı Çarşamba Perşembe Cuma Cumartesi Pazar

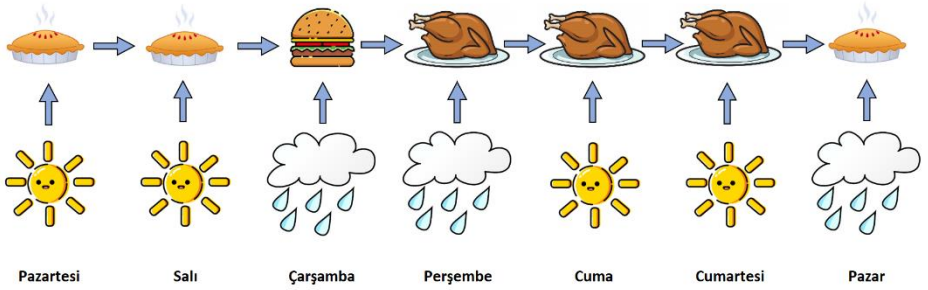


❖ Diğer kural, ***hava güneşliyse*** bahçede çalışıyor ve bir önceki günkü yemeğin aynısını, ***hava yağmurluysa*** yeni yemek yapıyor.

Perşembe => Yağmurlu Hava (Yemeği değiştir) + Hamburger = Tavuk

İki kuralı da aynı anda uygulayalım ve bir haftalık programı çıkartalım.





Perşembe => Yağmurlu Hava (Yemeği değiştir) + Hamburger = Tavuk

$h_{pazar} = f($  ,  $) =$ 

⚠ $h_{pazar} = f(h_{t-1}, x_t)$

RNN Avantajları

- Bir önceki örnek ile ilişki kurar ve girdiler unutulmadan ilerlenir.
- Kullanım alanı çok geniştir (*Metin ve ses verileri, sınıflandırma problemleri, regresyon problemleri, üretken (generative) modellerde kullanılır.*)

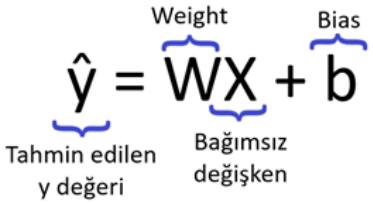
RNN Dezavantajları

- Gradient vanishing /exploding problemlerine sahiptir.
- Uzun girdileri işlemekte zorlanır.
- Kendi içerisinde de işlemler yapıldığı için, genelde çok derin RNN yapıları kurulmaz.

Kaynak : medium.com/deep-learning-turkiye

Terminoloji

Weight(Ağırlık)



Weight değeri bağımsız değişken ile çarpılıp, sonrasında bias ile toplanır ve tahmini sonuca (y) ulaşılır.

bias=0 alıp, hava sıcaklığı ile dondurma satışları arasındaki ilişkiyi inceleyelim.
Havanın 27° olduğunu ve hava sıcaklığının ağırlığının da 5 olduğunu kabul edelim.

Hava 27° 135 = 5 * 27 + 0

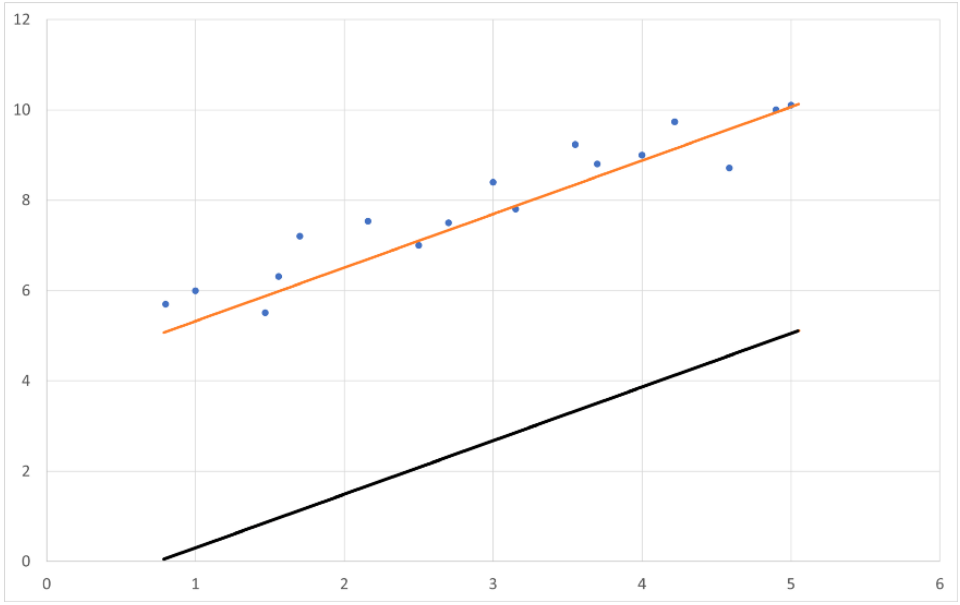
Hava 22° 110 = 5 * 22 + 0

5° derecelik fark satışları 25 adet dondurma aşağı çekmiştir.
1° derecelik değişim dondurma miktarını 5 birim arttırır veya azaltır.

Ağırlık aynı zamanda tahmin için çizilen doğrunun eğimidir.
135-110 = m (27-22) m=5

Bias

- x = 0 iken y'nin değerini alır.
- Özel bahçe ve havuzun ev fiyatı üzerindeki etkisini
- Özel bahçe ve havuzun ev fiyatı üzerinde etkisi yüksektir ancak bunlara sahip olmayan bir evin fiyatı da 0 değildir.
- Ev fiyatını etkileyen 1000 farklı özellik olabilir ama bizim elimizde sadece 2 tane özellik var.
- **Bias** ekleyerek, sonucumuza ekleyemediğimiz 998 özelliği temsil eden bir değer olmasını bekler ve sonucumuzu gerçek değere yaklaştırmasını bekleriz.



Turuncu, Bias eklenmiş halidir.

Overfitting

- Bir modelin x girdilerine karşılık gelen y çıktıları ile birlikte eğitildiğini kabul edelim. (x,y) ilişkisininin bir tahmini fonksiyonunu (Y) elde etmeye çalışalım.
- Bu fonksiyonun veri setine çok fazla uyması, veri setini ezberlemesi demektir (overfitting).
- Overfitting, her veriye tamamen uyacak şekilde bir fonksiyonun geliştirilmesi ve karmaşık fonksiyonlar üretebilmesi durumlarında ortaya çıkar.
- Modelden daha önce görmediği bir veriyi tahmin etmesi istendiğinde veri setini ezberlediğinden doğru tahmin yapamaz.

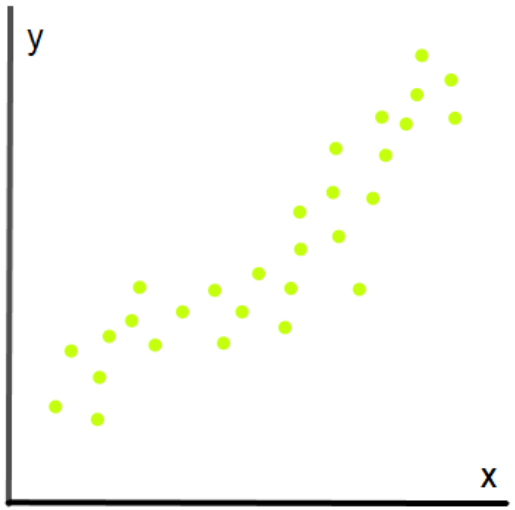
Örnek: Bir öğrenci sınava hazırlanırken konulara çalışmak yerine sadece önceki çıkmış soruları ezberler ve sınavda da daha önce hiç görmediği sorular geldiğinde başarısız olur.

Underfitting

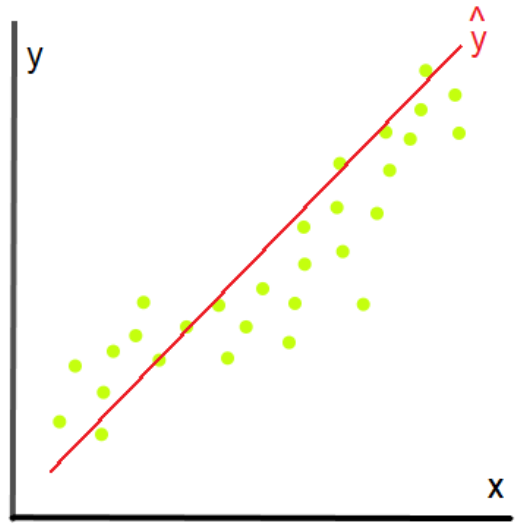
- Y tahmini fonksiyonunun veri setinden çok bağımsız olması yani veri setinin ilişkisini yansıtamaması durumudur (underfitting).
- Underfitting, modelin çok basit olması yani verinin yalnızca çok küçük bir kısmını ifade edebilecek yeterlilikte fonksiyon üretmesidir.

Örnek: Öğrenci eğer konuların sadece bir kısmını çalışırsa ve sınavda tüm konulardan sorumlu olacağı için başarısız.

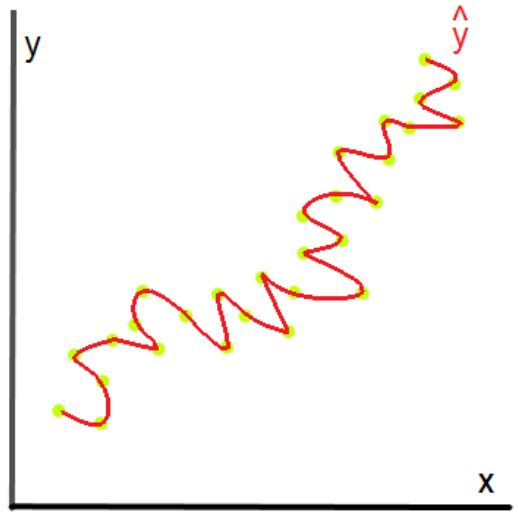
Modelden beklentimiz veri setini ezberlememesi ama aynı zamanda veri setinin geneline de hakim olmasıdır.



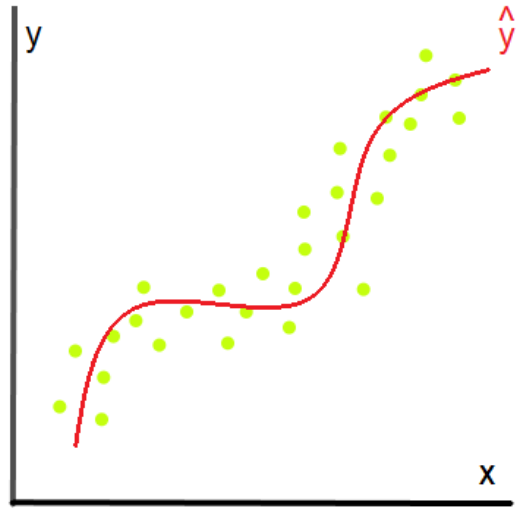
Veri Seti



Underfitting



Overfitting



İdeal Model

Hiperparametre

Veri setini hazırlarken, modeli oluştururken veya eğitirken kullanıcı tarafından belirlenmesi gereken bileşenler.

- KNN algoritmasında k nın değeri ?
- SVM algoritmasında kernel fonksiyonu ?
- Derin ağ modellerindeki learning rate, mini batch, epoch, gizli katman sayısı, kaç düğüm (node) içereceği, başlangıç ağırlıkları, aktivasyon fonksiyonu

Ne olması gerektiği, modeli tasarlayan kişiye bırakılmış, probleme, veri setine göre değişiklik gösteren parametrelere hiper-parametre denir.

- Modelin yüksek başarım sağladığı birbirinden farklı hiper parametre grupları olabilir.
- En uygun hiper parametre grubunun seçilmesi önemli bir problemdir.
- Tasarımcının sezgisine, önceki problemlerden elde edilen tecrübelere, farklı alanlardaki uygulamaların kendi problemimize yansımasına, güncel trendlere, vb. göre değişebilir.
- En uygun hiper parametre grubunun en optimum şekilde seçilmesine yönelik farklı teknikler (stochastic gradient descent, adagrad, adadelta, adam, adamax) kullanılır.

Derin Öğrenmede En Sık Kullanılan Hiperparametreler

Veri Setinin Boyutu

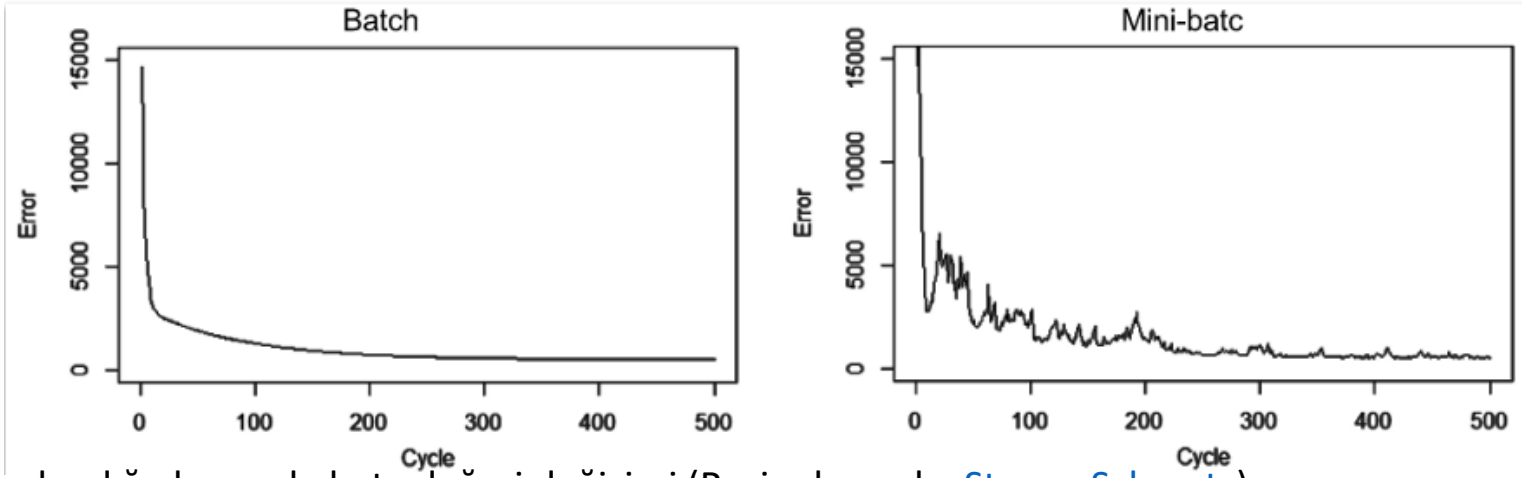
- Derin öğrenme uygulamalarında veri setinin büyüklüğü ve çeşitliliği öğrenme için önemlidir. Veri seti ne kadar büyük ise öğrenme o kadar iyi olur.
- Ancak, veri setinin büyüklüğü öğrenme için harcanan zamanı ve modelin büyüklüğünü de arttırır.
- Eğitimin sık yapılmayacağı ve depolama alanının sorun olmadığı uygulamalarda öğrenim başarısı tercih edilir.
- Eğitimin sık yapılacağı veya mobil ortamlar gibi depolama alanının problem olacağı uygulamalarda başarımın yanı sıra veri setinin büyüklüğü de değerlendirilmelidir.
- Veri setinin büyük olması her zaman iyi bir model için yeterli değildir, veri setindeki çeşitlilikte önemlidir.
- Veri seti büyüdükçe başarım sonsuza kadar doğru orantılı olarak artmaz, belli bir noktadan sonra artışlar küçük olur.
- Modeli eğitirken veri setinin hangi noktada kırıldığına da dikkat edilmelidir.
- Veri seti küçük ise sentetik veri ile arttırılır. Transfer Learning ile öznitelik transferi yapılır. Küçük veri setlerinde Non-lineer problemlerin çözümü için SVM tercih edilir.
- Görsel veriler üzerinde çalışılıyorsa her sınıf için binlerce örnek olmalıdır.
- Sınıflar birbirine çok benziyorsa, veri seti içerisinde geçişgenlik varsa «batch» değeri değiştirilir.

“Mini-Batch” Boyutu

- Modele eğitim sürecinde tüm veri setini girdi olarak vermek yerine, veriyi küçük parçalar halinde bölmemizi sağlayan hiperparametredir.
- Parametreler eğitim esnasında güncellenerek en optimal sonuca ulaşırken, güncelleme işlemi tüm veri seti işlendikten sonra gerçekleştirilirse hem zaman kaybına hem de işlem maliyetinin artmasına neden olur.
- Öğrenmenin her iterasyonunda geriye yayılım (“backpropagation”) işlemi ile ağ üzerinde geriye dönük olarak gradyan (“gradient descent”) hesaplaması yapılarak ağırlık değerleri güncellenir.
- Bu hesaplama işleminde veri sayısı ne kadar fazla ise hesaplama da o kadar uzun sürer.
- Bu yüzden veri seti parçalara ayrılarak, güncelleme işleminin yapılması sağlanır (parçaların büyüklüğü de **batch_size** olarak adlandırılır).
- Batch_size belirlenirken veri setinin büyüklüğü, verinin dağılımı ve makinenin işlem gücü dikkate alınmalıdır.
- batch_size=1 seçilirse, her veri sonrası güncelleme yapılır (schoastic gredient decent). Sırasıyla gelen veriler birbirlerinden çok farklı ise tahmin sonuçlarımız da o kadar farklı olur. Bu istenmeyen bir durumdur. batch_size= toplam veri sayısı seçilirse (batch gredient descent), güncelleme sayısı artacağından makinenin işlem gücü yetersiz kalır (batch_size a değer atanmazsa otomatik olarak batch_size=toplam veri sayısı olarak alınır).

schoastic gredient decent mini-batch=1	**batch gredient descent** mini-batch=tüm veri seti
▪ Modelin gürültüyü öğrenmesine neden olabilir. Grafikteki zikzaklar artar. Öğrenme sürecinde her defasında farklı veri kullanılır. ▪ Local optimum'da takılıp, global optimuma hiç ulaşamayabilir. ▪ Salınımı azaltacak yöntemler kullanılır RMSprob (Root Mean Square error Propability).	▪ Model daha az gürültü öğrenecektir. ▪ Bütün veriyi işlediği için öğrenme çok uzun sürecektir. ▪ Optimizasyon işleminde büyük adımlarla ilerleme olacaktır.

Batch seçiminde verilerin rastgele seçilmesi önemlidir.



Batch ve mini-bath uygulandığı durumda hata değeri değişimi (Resim kaynak : Steven Schmatz)

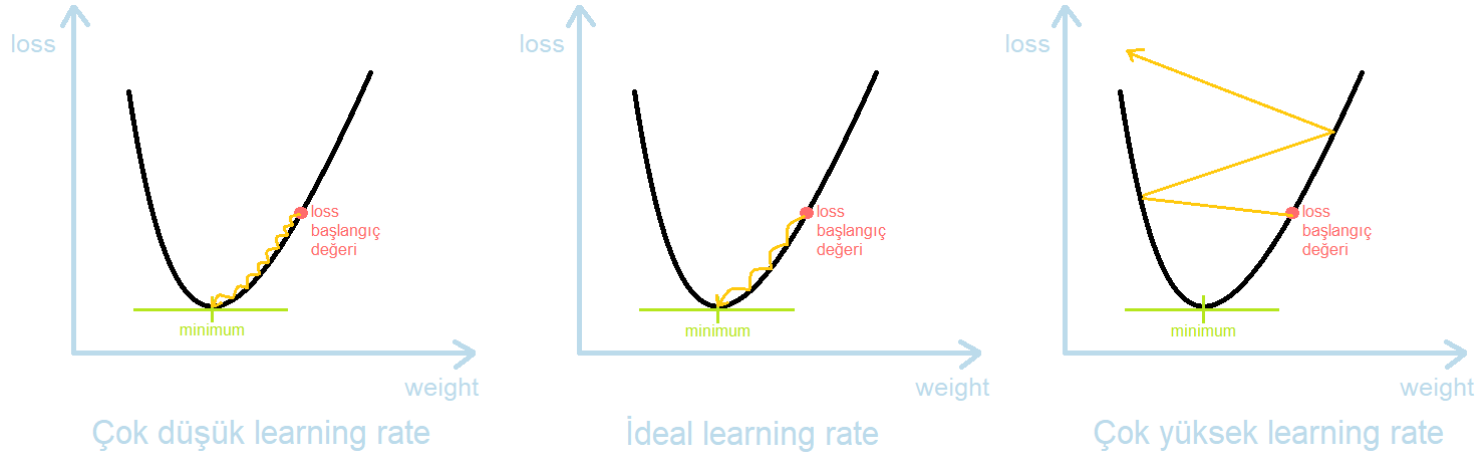
Epoch-İterasyon

- Tüm verilerin ağ tarafından bir kere işlenmesine 1 epoch denir.
- Verileri tek seferde işlemeyip, mini-batch'lere böldüğümüzde, her bir mini-batch 1 iterasyona karşılık gelir.

20 kıtadan oluşan bir şiir olsun (20 kıta = 1 epoch) ve ezberlerken 2 kıtaya bölüp ezberleyelim (2 kıta = mini-batch) böylece (iterasyon sayısı=10) olur.

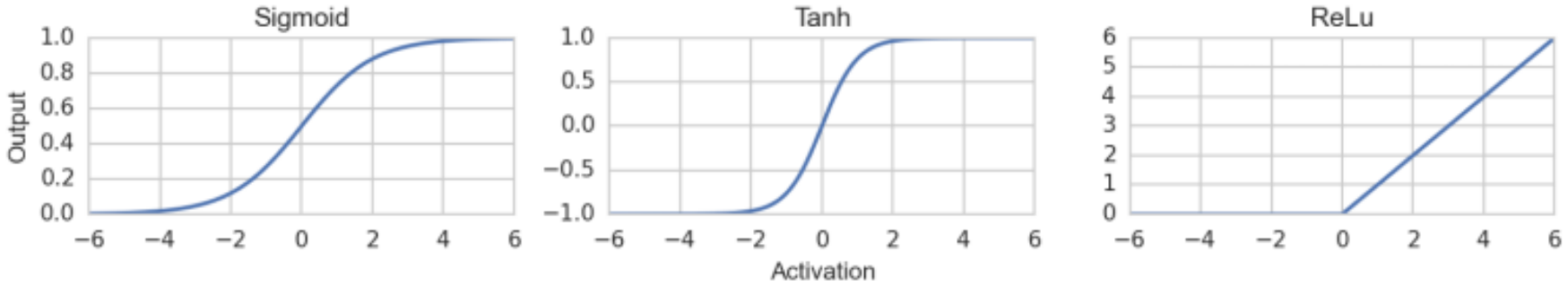
Learning Rate

- Kayıp fonksiyonu üzerinde optimum noktaya (kayıp değerinin minimum olduğu) ulaşmak için gradyan iniş kullanılır. Bu iniş esnasında izlenen yolda atılacak adım büyüklüğü **learning rate** olarak tanımlanır.
- Learning rate, eğitimden önce belirlenen bir hiperparametredir.
- Çok küçük seçilirse eğitim için gerekli epoch sayısı artar ve eğitimin süresi uzar.
- Çok büyük seçilmesi minimum noktanın atlanmasına ve optimuma ulaşılamayacağı anlamına gelir.
- Bu gibi durumların önlenmesi için ideal bir learning rate değeri seçilmelidir.
- Learning rate için varsayılan değer 0.01, belli bir epoch'dan sonra da 0.001'e kullanılır.



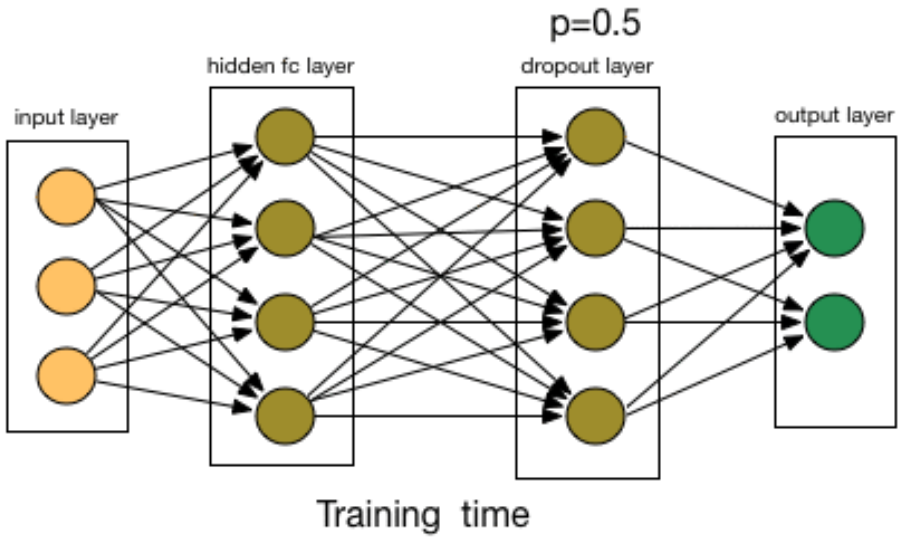
Aktivasyon Fonksiyonu

- Derin öğrenme yöntemleri doğrusal olmayan (non-linear) yapıya sahip problemlerin çözümünde kullanılır.
- Gizli katmanlarda y = f(x,w) şeklindeki lineer fonksiyonumuzda matris çarpımı yapılıp, nöronların ağırlığı hesaplandıktan sonra, çıktı doğrusal olmayan (non-linear) bir değere dönüştürülür.
- Sonucun non-linear hale dönüştürülmesini aktivasyon fonksiyonları sağlar.
- Gizli katmanlarda geri türev alınabilmesi (gradient decent hesaplanabilmesi) için (öğrenmede fark, geri türevle alınır) gizli katmanların çıktısı bazı aktivasyon fonksiyonları ile normalize edilir.
- Aktivasyon fonksiyonlarının bazıları sigmoid, ReLu, PreLu, …
- **ReLu'da parametreler sigmoid'e göre daha hızlı öğrenir.**
- PReLu ise, ReLU'nun kaçırdığı negatif değerleri yakalar (**bizim için negatif değerler önemliyse PReLu tercih edilmelidir).**



Dropout (Seyreltme) Değeri

- Tam bağlı katmanlarda belli eşik değerin altındaki düğümlerin çıkarılması başarımı arttırabilir.
- Zayıf bilgilerin unutulması, öğrenmeyi hızlandırabilir.



- Seyreltme (Dropout) eşik değeri olarak [0,1] aralığı ve genelde de 0.5 kullanılmaktadır.
- Probleme ve verisetine göre değişiklik gösterebilir.
- Seyreltme (Dropout) için rastgele eleme yöntemi de kullanılabilir.
- Tüm katmanlarda aynı dropout değerini kullanmak zorunlu değildir.

Train (Eğitim) Veri Seti

- Modeli eğitmek için kullandığımız veri setidir.

Validation (Doğrulama) Veri Seti

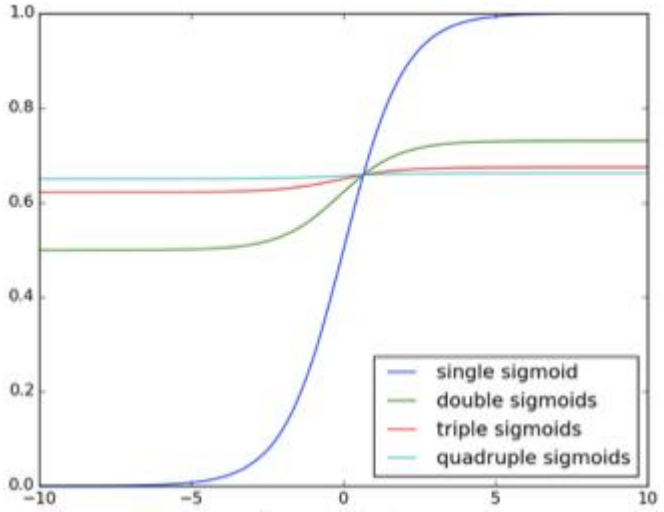
- Modelin hiperparametreleri ayarlanırken eğitim veri kümesine uyan bir modelin tarafsız olarak değerlendirmesini yapmak için kullanılan veri kümesidir.
- Sistemin uygun parametrelerini seçmek için kullanılır. Doğrulama seti, eğitim setinin bir parçası olarak kabul edilebilir çünkü modelinizi, sinir ağlarını veya diğerlerini oluşturmak için kullanılır.
- Bu iki veri setini, modelin doğruluğunu (accuracy) ölçmek için kullanmamalıyız.

Test Veri Seti

- Modelin doğruluğu daha önce hiç görmediği bir veri seti ile test edilir ki bu da Test veri setidir.

Vanishing/Exploding Gradient (Gradient yok olması/büyümesi)

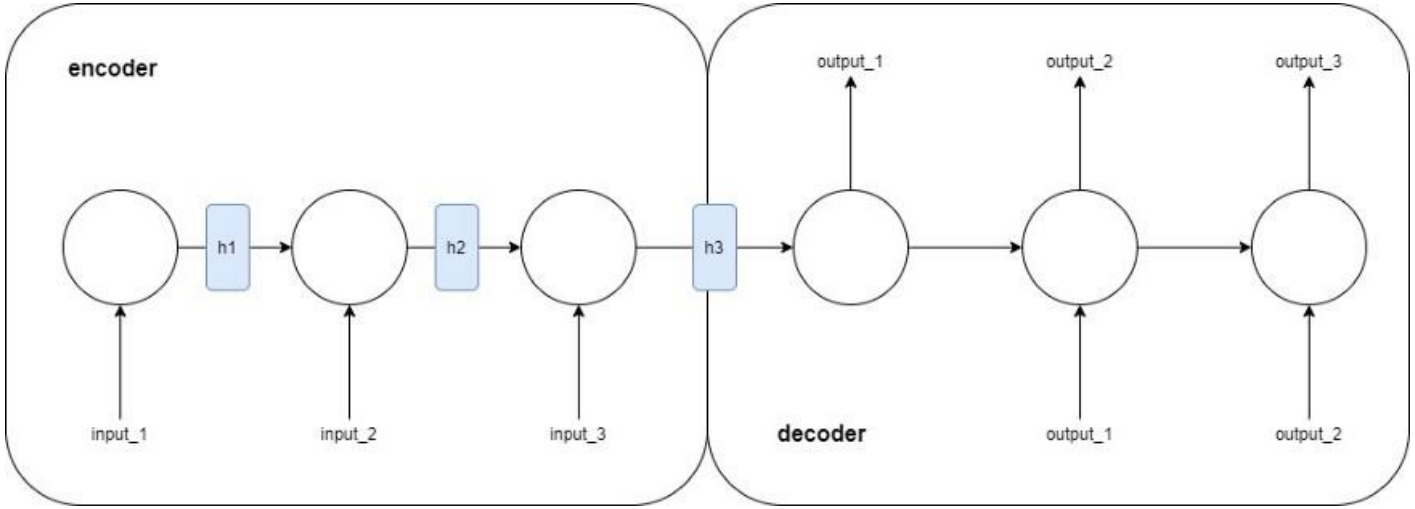
- **Gradient,** tüm ağırlıkları ayarlamamızı sağlayan bir değerdir.
- Birbirine bağlı uzun ağlarda hatanın etkisi gittikçe küçülür ve gradient kaybolmaya başlayabilir.
- Bütün katmanlar ve zamana bağlı adımlar birbirine çarpımla bağlı olduğundan, türevleri yok olma veya büyümeye meyillidir.
- Gradient yok olması/büyümesi ağın büyük değerler üretmesini sağlar ve bizi doğru sonuçtan uzaklaştırır. Eşik değer (Threshold) koyarak çok yüksek değerli gradientler atılır.
- Gradientler aşırı küçülerek yok olması durumunu tespit etmek zordur. Ne zaman durdurulması gerektiğini tahmin etmek zordur. Bazı çözümler mevcuttur.
  - W için uygun başlangıç değerleri seçmek
  - Sigmoid ve Tanh aktivasyon fonksiyonları yerine ReLU kullanmak
    ReLU fonksiyonunun türevi 0 veya 1'dir
  - Bu problemi çözmek için LSTM dizayn edilmiştir.



- RNN’ler Encoder-Decoder mimarisini kullanır. Encoder ve Decoder ayrı birer RNN’dir.
  Not: Kelimeler ile matematiksel işlemlerin yapılabilmesi için, kelimelerin WordEmbedding oluşturulur.

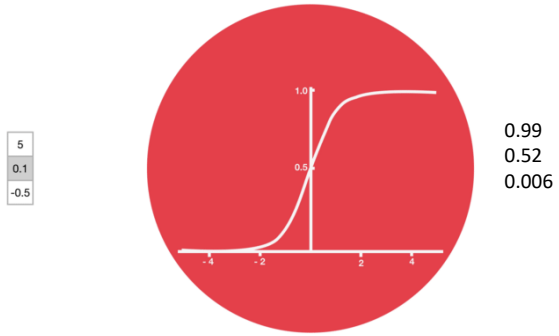
- Encoder kısmındaki RNN modeli eğitildiğinde, çıktı olarak üretilen Hidden Layer, Decoder bileşenine aktarılır. Bütün sistemi tek bir parça olarak düşünecek olursak Encoder’dan gelen bu Hidden Layer’ı, Decodar’daki ilk Hidden Layer olarak ve gelen bütün verinin özeti olarak da düşünebiliriz.

- Decoder’dan üretilen çıktı da probleme göre değişkenlik gösterir (girdi cümlenin başka bir dile çevirisi, kategorisi veya soru ise cevabı olabilir).



LSTM - Long Short Term Memory - Uzun Kısa Süreli Hafıza

- RNN'ler kısa cümleler üzerinde iyi sonuçlar verse de uzun cümleler/paragraflarda başarılı değildir.
- LSTM'ler uzun vadede bağımlılıkları öğrenebilen RNN'nin özel bir türüdür.
- LSTM, 1977 yılında *vanishing gradient* problemini çözmek için geliştirilmiştir.
- LSTM, duygu analizi, metin üretme ve zaman serileri gibi birçok konuda kullanılır.



Vanishing Gradient Problemi

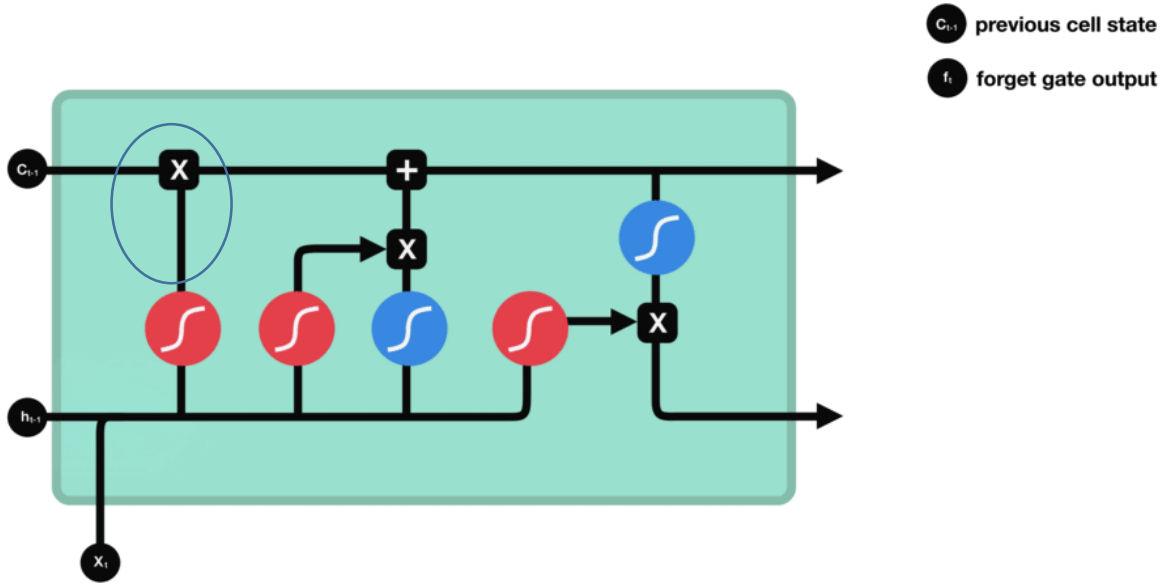
- Aktivasyon fonksiyonları sayesinde girdimizi -1 ve 1 veya 0 ve 1 aralığına indiririz.
- Dar bir aralığa indirgediğimiz için girdimizdeki büyük bir değişim aktivasyon fonksiyonunda çok büyük bir değişime yol açmaz.
- Bu sebeple türevi küçük olur ve o katman yeteri kadar öğrenemez.
- RNN'de çok erken aşamalarda bu durum ile karşılaşıldığından uzun metinlerde gördüklerini unutabilir ve böylece kısa süreli bir hafızaya sahip olurlar.

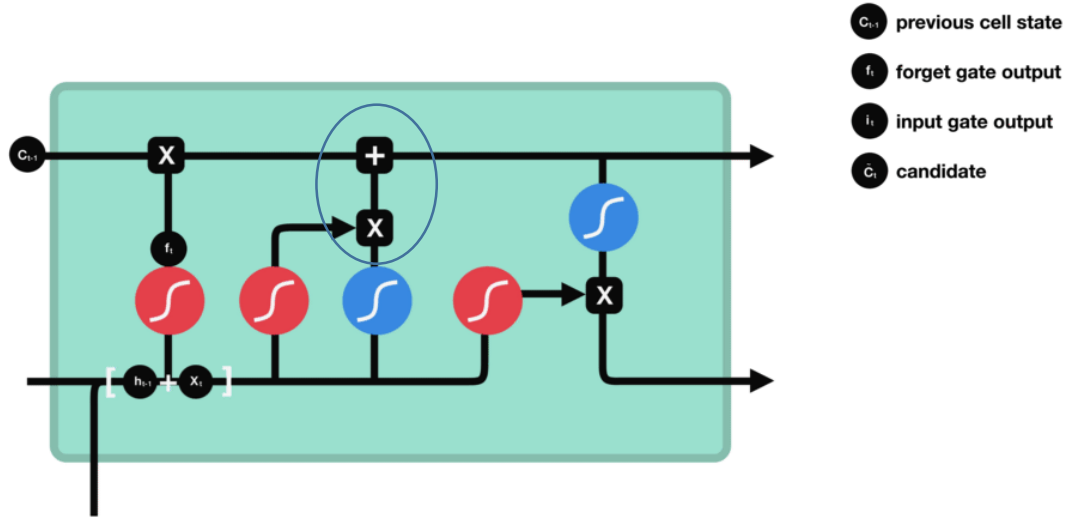
LSTM ile bu sorun nasıl çözülüyor?

- LSTM'in iç yapısında kapılar (gate) mevcuttur.
- Neyin hatırlanacağını, neyin unutulacağını bu kapılar belirler.
- Gelen girdi önemsizse unutulur, önemliyse bir sonraki aşamaya aktarılır.
- ***Gate*** (input, output, forget) ve ***Cell State***

Forget Gate (Unutma Kapısı)

- Hangi bilginin tutulacağı veya unutulacağına karar verir.
- Bir önceki gizli katmandan gelen bilgiler(ht-1) ve güncel bilgiler(xt) Sigmoid Aktivasyon fonksiyonundan geçer. Değer 0'a ne kadar yakınsa o kadar unutulacak, 1'e ne kadar yakınsa o kadar tutulacak (cell state taşınacak) demektir.



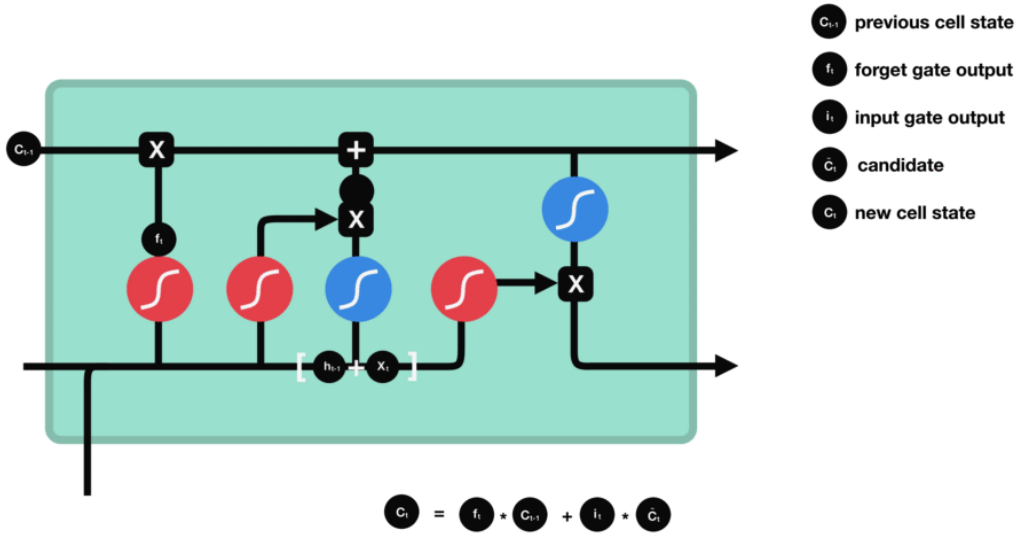


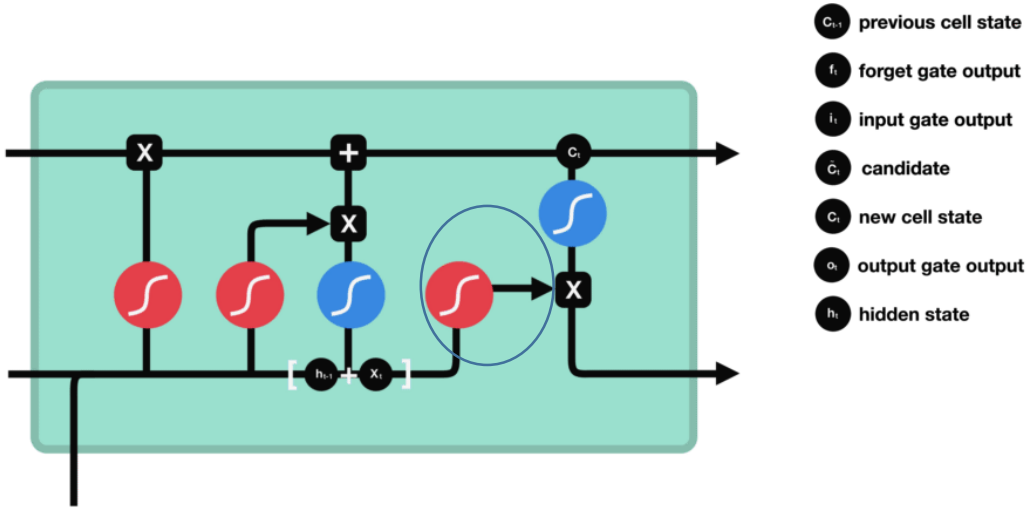
Input Gate (Girdi Kapısı)

- Cell State'i güncellemek için kullanılır. Sigmoid fonksiyonu uygulanır, hangi bilginin tutulacağına karar verilir.
- Ağı düzenlemek için veriyi -1 ve 1 aralığına sıkıştıran Tanh aktivasyon fonksiyonu kullanır. Sigmoid ve Tanh fonksiyon çıkışları çarpılır ve hangi bilginin güncelleneceği kararına varılır.

Cell State

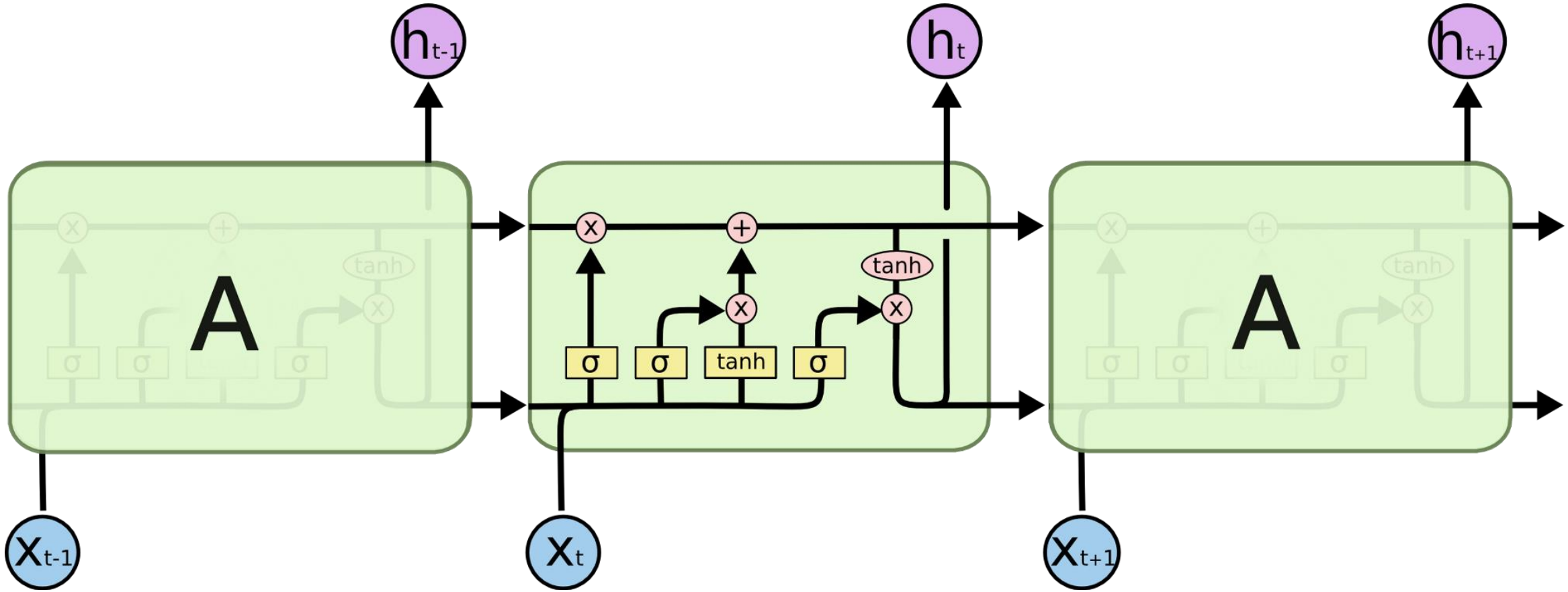
- Cell State'in hücre içerisindeki en önemli görevi tahmin yapmak için anlamlı bilgiyi hücreler boyunca taşımaktır.
- Taşınması gereken verileri alır ve hücre sonuna, oradan da diğer hücrelere taşır.
- İlk olarak Forget Gate'den gelen sonuç ile bir önceki katmanın sonucu çarpılır, sonra Input Gate'den gelen değer ile toplanır.





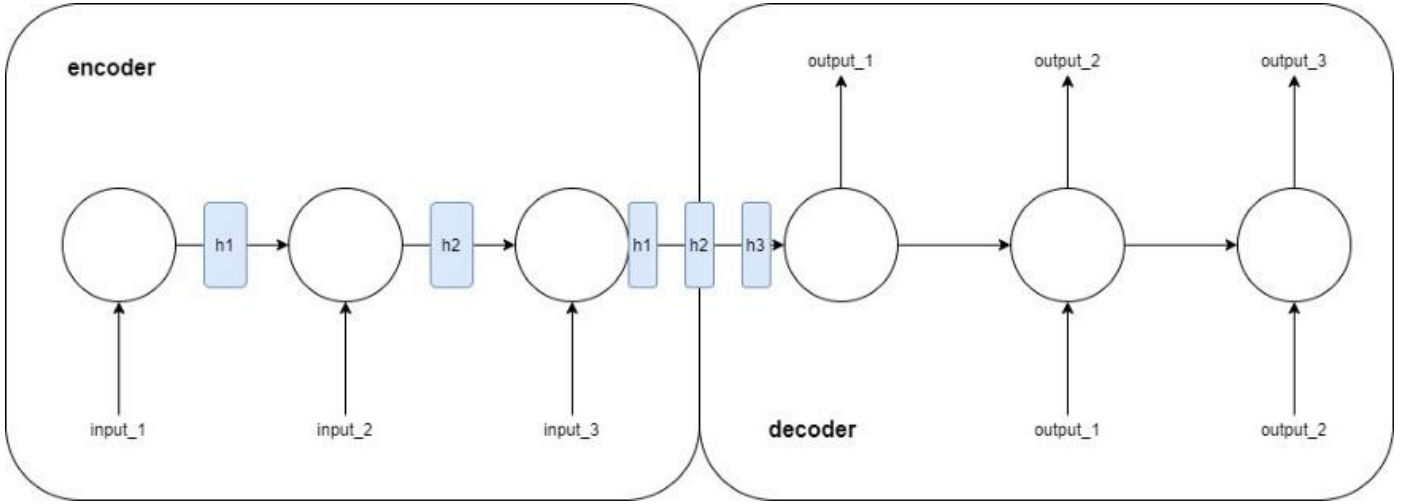
Output Gate (Çıktı Kapısı)

- Bir sonraki katmana gönderilecek değere karar verir. Bir sonraki hücrenin girişini (ht+1) belirler.
- Bir önceki girdi ile şu anki girdi bilgisi Sigmoid fonksiyonundan geçer.
- Cell State'den gelen değer, Tanh fonksiyonundan geçtikten sonra iki değer çarpılır ve bir sonraki katmana "Bir önceki değer" olarak gider. Cell State ilerler.


h_{t-1}
h_t
h_{t+1}
A
A
tanh
σ
σ
tanh
σ
X_{t-1}
X_t
X_{t+1}

Attention Mekanizması

- Encoder'daki bütün bilginin sabit uzunluktaki bir vektörle ifade edilmesi ile hatırlama problemi kısmen de olsa ortadan kalkıyor.
- Attention mekanizması, geleneksel RNN mimarisindeki gibi sadece en son Hidden Layer'ı Decoder'a göndermek yerine, bütün oluşan Hidden Layer'ları bir arada Decoder'a gönderir.
- Verideki ilk kelimelerin önemi, son kelimelerde olduğu gibi korunur ve bilgi bütünlüğü daha iyi korunur. Decoder'da, her bir adımda oluşturulan Hidden Layer'ların oluşturduğu matrix'ten, o adım için bir vektör oluşturulur.
- Bu vektör Decoder'daki Hidden Layer'la bir arada işlenerek o adımın çıktısı oluşturulur.



Özet

- RNN, gelen kelimeleri sırayla değerlendirdiği için kelimelerin bütünlüğünü korur. Ancak, girdi uzunsa, kelimeler arasındaki ilişkiyi unutur ve cümle başındaki kelimeler gerçekte olduğundan çok daha önemsiz olarak modellenebilir.
- Attention, her kelimenin işlenmesinden sonra oluşan Hidden Layer bilgisini decodera aktardığı için RNN'de görülen, baştaki kelimelerin değerinin azalması problemini az da olsa çözmeye çalışır.
- Transformer, paralel hesaplama yapmakla ön plana çıkar. Paralel işlem yapmanın yanı sıra, multi-head attention mekanizmasına sahip olduğundan, gelen veriyi soldan-sağa/sağdan-sola değerlendirdiği için, eğitimin sonunda içeriğe daha fazla hakim olur.

https://stanford.edu/~shervine/l/tr/teaching/cs-230/cheatsheet-recurrent-neural-networks#