Deep Learning

Neural network'lerin nasıl yapılandırıldığı -- katmanlar, ağırlıklar, aktivasyon fonksiyonları -- ve backpropagation ile eğitim, ve CNN/RNN/transformer mimarilerine kısa bir bakış.

Başlangıç 18 dk

Son güncelleme: 6 Ekim 2026
Çevrimiçi oku: https://www.learnforgex.com/tr/topics/deep-learning

---

Deep Learning

"Machine Learning" dersi, yeterince veri ve hesaplama gücü mevcut olduğunda diğerlerinden çarpıcı biçimde daha iyi ölçeklendiği ortaya çıkan belirli bir öğrenme algoritmaları ailesine işaret ederek bitmişti: **neural network'ler (sinir ağları),** özellikle *çok katmanlı* olanları. Bu aile Deep Learning'dir -- "Yapay Zeka Nedir?" dersindeki haritanın üçüncü dairesi, Machine Learning'in içinde, o da AI'ın içinde yer alır. Deep Learning, önceki derste öğrendiğiniz hiçbir şeyin (training, inference, overfitting, training/test ayrımı) yerini almaz -- bunların hepsini gerçekleştirmek için belirli, son derece başarılı bir tekniktir. Bu ders kavramsal kalıyor ve bilinçli olarak altındaki matematikten kaçınıyor; amaç, bir neural network'ün nasıl yapılandırıldığına ve "derinliğin" neden önemli olduğuna dair doğru bir zihinsel model kurmak, arkasındaki kalkülüs değil.

Deep Learning Nedir?

Deep Learning, veriden öğrenme mekanizması olarak **neural network'leri** -- biyolojik nöronların birbirine nasıl bağlandığından gevşek biçimde ilham alan yapılar -- kullanan bir Machine Learning alt kümesidir. "Deep" (derin), özellikle üst üste yığılmış birçok katmana sahip ağları ifade eder (bir sonraki bölümde katmanın tam olarak ne olduğunu göreceksiniz). Yalnızca bir ya da iki katmanlı "sığ" (shallow) bir neural network 1950'lerden beri var; *deep* learning'i farklı kılan ve bu derse adını veren şey, bu tür birçok katmanı bir araya yığmaktır -- bu, bir ağın sığ bir ağın öğrenebileceğinden çok daha karmaşık örüntüler öğrenmesini sağladığı ortaya çıkan bir şeydir.

Nöronlar, Katmanlar ve Ağlar: Temel Yapı Taşları

Bir neural network, genellikle "nöron" ya da "node" (düğüm) denen birçok basit birimden oluşan üç tür katmandan inşa edilir:

- **Input layer (girdi katmanı):** bir örneğin ham feature'larını alır ("Machine Learning" dersindeki "feature"ları hatırlayın) -- bir görüntü için bu, her pikselin parlaklık değeri olabilir; bir cümle için, her kelimenin sayısal bir temsili.
- **Hidden layer'lar (gizli katmanlar):** girdi ve çıktı arasında bir ya da daha fazla katman, asıl öğrenmenin gerçekleştiği yer. Gizli katmandaki her node, önceki katmanın çıktılarını alır, birleştirir ve yeni bir değeri ileri geçirir. Bir ağa "deep" (derin) denmesinin özel nedeni, sıralı olarak yığılmış birçok bu tür gizli katmana sahip olmasıdır.
- **Output layer (çıktı katmanı):** ağın nihai cevabını üretir -- bir görüntünün kedi olma olasılığı, tahmin edilen ev fiyatı, bir cümledeki sıradaki kelime.

Veri, ağ boyunca girdiden, sırayla her gizli katmandan geçerek çıktıya akar -- buna **forward pass (ileri geçiş)** denir, ve ağ zaten eğitildikten sonra inference sırasında (önceki dersteki "Training ve Inference"e bakın) olan şey tam olarak budur.

Her Node Gerçekte Ne Yapar?

Altındaki matematiğe girmeden, tek bir node'un içinde ne olduğunun şeklini bilmek yardımcı olur: önceki katmandan gelen değerleri alır, bunları **weight** (ağırlık) denen ayarlanabilir sayılarla birleştirir (iki node arasındaki her bağlantının kendi ağırlığı vardır, kabaca "bu girdi ne kadar önemli"yi temsil eder), genellikle buna node'un kendine ait, girdilerden bağımsız sabit bir **bias** değerini de ekler, ve sonucu **activation function (aktivasyon fonksiyonu)** denen küçük bir matematiksel fonksiyondan geçirir. ("Parametre" dendiğinde kastedilen yalnızca ağırlıklar değildir -- weight'ler ve bias'lar birlikte bir ağın parametreleridir; bu ders bias'ın ayrıntısına girmiyor, yalnızca var olduğunu bilmeniz yeterli.) Aktivasyon fonksiyonunun görevi, ağın *non-linear* (doğrusal olmayan) örüntüleri temsil edebilmesini sağlamaktır -- girdilerin basit, düz-çizgi bir birleşimiyle yakalanamayan ilişkiler, ki bu neredeyse her ilginç gerçek dünya örüntüsünü tanımlar ("bu bir kedi fotoğrafı mı" için düz-çizgi bir formül yoktur). Bu seviyede deep learning'i anlamak için belirli aktivasyon fonksiyonlarının formüllerini bilmenize gerek yok -- yalnızca bu adımın, bir ağın "girdilerinin ağırlıklı bir toplamından" çok daha fazlasını öğrenmesini sağladığını bilmeniz yeterli.

Somut, çok küçük bir örnek (matematik öğretmek için değil, yalnızca "girdi + ağırlık + aktivasyon" mantığını gözünüzde canlandırmak için): bir node iki girdi alsın, `x1 = 2` ve `x2 = 1`, ağırlıkları da sırasıyla `w1 = 3` ve `w2 = -1` olsun. Node önce ağırlıklı toplamı hesaplar: `(2 × 3) + (1 × -1) = 5`. Sonra bu değeri bir aktivasyon fonksiyonundan geçirir -- basitleştirilmiş bir örnek olarak yaygın kullanılan ReLU'yu (negatifse 0'a yuvarlar, pozitifse değiştirmez) alırsak, 5 zaten pozitif olduğu için sonuç yine 5'tir. Gerçek ağlarda tek fark ölçektir: binlerce node, milyonlarca ağırlık -- ama her birinin yaptığı iş, bu birkaç satırlık işlemin ta kendisidir.

Bir Ağ Nasıl Öğrenir: Loss ve Backpropagation

Bir neural network'ü eğitmek, "Machine Learning" dersindeki aynı training/ inference ayrımını izler, eğitim yarısı için belirli bir mekanizmayla:

1. Ağ, bir eğitim örneği üzerinde bir forward pass yapar ve bir tahmin üretir.
2. Bir **loss function (kayıp fonksiyonu),** o tahminin doğru label'a (önceki dersteki "label" ları hatırlayın) kıyasla ne kadar yanlış olduğunu ölçer -- düşük olması "doğruya daha yakın" anlamına gelen tek bir sayı.
3. **Backpropagation (geri yayılım),** ağ boyunca geriye doğru çalışarak, her bir ağırlığın o hataya ne kadar katkıda bulunduğunu hesaplar.
4. Her ağırlık, hatayı azaltacak yönde hafifçe itilir -- **gradient descent (gradyan inişi)** denen bir süreç.
5. Bu, birçok örnek boyunca tekrar tekrar gerçekleşir, ta ki ağın tahminleri güvenilir biçimde iyileşene kadar.

Kavramsal olarak, eğitim döngüsünün tamamı budur. Bu kurs, backpropagation'ın arkasındaki kalkülüsü bilinçli olarak türetmiyor -- akılda tutulması gereken önemli fikir, derin bir ağı eğitmenin bu tahmin-ölç-hata-ayarla döngüsünü milyonlarca kez tekrarlamak anlamına geldiği, ve tam olarak bu yüzden training'in yavaş, hesaplama açısından pahalı aşama olduğu (önceki dersteki "Training ve Inference"e bakın) ve neden özel donanıma (tam olarak bu tür tekrarlı sayısal hesaplama için üretilmiş GPU'lara) bu kadar bağımlı olduğudur.

Neden "Deep"? Birçok Katmanın Rolü

Bir ağdaki her katman, kendinden önceki katman üzerine inşa ederek, girdinin giderek daha soyut bir temsilini öğrenme eğilimindedir. Görüntü tanımadan klasik, sezgisel bir örnek: erken katmanlar basit kenarları ve renk geçişlerini tespit etmeyi öğrenebilir; orta katmanlar bu kenarları kavis ve köşe gibi şekillere birleştirir; sonraki katmanlar şekilleri tanınabilir parçalara (bir göz, bir kulak) birleştirir; son katmanlar parçaları bütün kavramlara ("kedi") birleştirir. Hiç kimse bir katmanı açıkça kenar ya da kulak aramaya programlamaz -- bu temsil hiyerarşisi, yukarıda anlatılan eğitim sürecinden doğal olarak ortaya çıkar. Birçok katmanı üst üste yığmak, bu kademeli soyutlamayı mümkün kılan şeydir; yalnızca bir ya da iki katmanlı sığ bir ağın, ham piksellerden "kedi" kadar soyut bir kavrama kadar yükselecek yeterli adımı basitçe yoktur. Bu kenar→şekil→göz/kulak→kedi anlatımının *sezgisel* bir açıklama olduğunu, gerçek bir ağın belirli bir katmanının kesinlikle bu şekilde insan tarafından adlandırılabilir bir kavramı öğrendiğinin garanti edilmediğini de belirtmekte fayda var -- katmanlar genellikle gerçekten kademeli, soyutlanan örüntüler öğrenir, ama bunların tam olarak "kenar," "şekil" ya da "kulak" gibi temiz insan kategorilerine denk düşmesi varsayılmamalıdır.

> Bir modelin "milyarlarca parametresi olduğunu" duyduğunuzda, o parametreler neredeyse tamamen "Her Node Gerçekte Ne Yapar?" bölümünde anlatılan ağırlıklar ve bias'lardır -- çok derin, çok geniş bir ağın her katmanı boyunca, iki nöron arasındaki her bağlantı ve her node için bir sayı. Daha fazla parametre kabaca karmaşık örüntüleri temsil etme kapasitesinin daha fazla olduğu anlamına gelir, ama bunları iyi eğitmek için de daha fazla veri ve hesaplama gerektiği anlamına gelir.

Yaygın Neural Network Türleri

Farklı network *mimarileri* -- katmanları ve bağlantıları düzenlemenin farklı yolları -- farklı veri türlerine uygundur. Yalnızca kavramsal düzeyde bile olsa, isimleriyle bilinmeye değer üç tanesi var, çünkü bu kursta ilerleyen bölümlerde tekrar karşınıza çıkacaklar:

- **Convolutional Neural Networks (CNN'ler):** görüntüler gibi grid-benzeri veriler için özelleşmiştir -- katmanları, görüntünün neresinde olurlarsa olsunlar yerel örüntüleri (önceki bölümdeki kenarlar ve şekiller gibi) tespit edecek şekilde yapılandırılmıştır.
- **Recurrent Neural Networks (RNN'ler):** sıralı veri (metin, ses, zaman serisi) için tasarlanmış, tarihsel olarak önemli bir mimari -- bir ağın kendinden önce gelenin bir tür "hafızasını" tutarak elemanları birer birer işlediği bir yapı. Transformer'lar yaygınlaşmadan önce dil görevleri için standart yaklaşımdı, bugün büyük ölçüde onların yerini almış durumda.
- **Transformer'lar:** daha yeni bir mimari (2017'de tanıtıldı), token'lar arasındaki ilişkileri -- girdinin her parçasının her diğer parçası için ne kadar önemli olduğunu -- *attention* denen bir mekanizmayla modeller. RNN'lerin aksine bu ilişkileri eleman eleman sırayla değil, eğitim sırasında büyük ölçüde paralel biçimde hesaplayabilir, bu da devasa veri kümelerinde çok daha verimli ölçeklenmelerini sağlar. Transformer'lar, bu kursta ilerleyen bölümlerde ele alınan hemen hemen her büyük dil modelinin (large language model) arkasındaki mimaridir -- bu ders nasıl çalıştıklarına daha derin girmeyecek, ama ismini şimdiden bilmeye değer, çünkü "Büyük Dil Modelleri" kategorisi doğrudan bunun üzerine inşa edilecek.

Deep Learning 2010'larda Neden Patladı?

"Yapay Zeka Nedir?" dersi, büyük etiketli veri kümeleri ve güçlü GPU'lar mevcut olduğunda deep learning'in 2012 civarında eski tekniklerden çarpıcı biçimde daha iyi performans göstermeye başladığından bahsetmişti -- bu ders artık bu iki şeyin neden bu kadar önemli olduğunu açıklayabilir. Birçok katmana ve milyonlarca (ya da milyarlarca) ağırlığa sahip derin ağların, overfitting yapmak yerine (önceki dersteki "Overfitting ve Underfitting"e bakın) güvenilir örüntüler öğrenmesi için devasa miktarda eğitim verisine ihtiyacı vardır, ve 2000'ler ile 2010'larda ortaya çıkan internet-ölçeğindeki veri kümeleri tam olarak bunu sağladı. Bu arada, "Bir Ağ Nasıl Öğrenir: Loss ve Backpropagation" bölümünde anlatılan tekrarlı sayısal hesaplama -- milyonlarca ağırlık boyunca milyonlarca küçük ayarlama -- GPU'ların (aslen video oyunu grafiklerini render etmek için üretilmiş) son derece iyi olduğu tam olarak o tür paralel hesaplamadır. Hiçbir bileşen tek başına yeni değildi; 2012 civarında değişen şey, ikisinin de yeterince büyük ölçekte AYNI ANDA mevcut olmasıydı.

Best Practices

- Varsayılan olarak deep learning'e başvurmayın. İyi performans göstermesi için genellikle diğer machine learning tekniklerinden ("Machine Learning"e bakın) çok daha fazla veri ve

hesaplamaya ihtiyaç duyar -- daha küçük ya da daha basit problemler için, deep-learning-olmayan bir yaklaşım inşa etmesi daha hızlı, çalıştırması daha ucuz ve anlaması daha kolay olabilir.
- Bir mimari seçerken, veri türüyle eşleştirin: grid-benzeri/görüntü verisi için CNN'ler, sıralı /metin verisi için RNN'ler ya da transformer'lar ("Yaygın Neural Network Türleri"ne bakın) -- verinize yanlış şekilde bir ağ kullanmak genellikle daha fazla çaba için daha kötü sonuçlar demektir.
- Önceki dersteki "Overfitting ve Underfitting"deki her şeyin hâlâ geçerli olduğunu, hatta çoğu zaman *daha fazla* geçerli olduğunu unutmayın -- derin ağlar devasa sayıda ağırlığa sahiptir, bu da veri çok küçük ya da tekrarlı olduğunda eğitim verisini basitçe ezberleme kapasitelerini artırır.

Yaygın Hatalar

- **"Deep learning" ile "AI"ı birbirinin yerine kullanılabilir saymak.** "Yapay Zeka Nedir?"teki haritanın gösterdiği gibi, deep learning machine learning'in içine, o da AI'ın içine yerleşmiş belirli bir tekniktir -- hiç neural network içermeyen, yararlı, köklü AI teknikleri de mevcuttur.
- **Bir ağın gizli bir katmanın ne yaptığını "anladığını" düşünmek.** "Neden 'Deep'? Birçok Katmanın Rolü"nde anlatılan katmanlı temsiller, eğitimden otomatik olarak ortaya çıkar -- hiç kimse belirli bir katmanı elle "kulak" tespit etsin diye tasarlamaz, ve belirli bir katmanın sonunda neyi kodladığı genellikle doğrudan insan tarafından okunabilir değildir.
- **Daha fazla katmanın ya da daha fazla parametrenin her zaman daha iyi bir model demek olduğunu varsaymak.** Belirli bir noktadan sonra, yeterli eşleşen veri olmadan daha fazla kapasite, gerçek dünya performansını artırmak yerine çoğunlukla overfitting riskini artırır (önceki derse bakın).

Özet, Cheat Sheet ve Terimler Sözlüğü

Özet

- Deep Learning, birçok üst üste yığılmış gizli katmana sahip neural network'ler kullanan Machine Learning'dir -- "deep" (derin), o katman yığınına işaret eder.
- Bir ağ, bir input layer, bir ya da daha fazla hidden layer, ve bir output layer'dan inşa edilir; veri, ağ boyunca bir forward pass ile hareket eder.
- Her node, girdilerini ayarlanabilir ağırlıklar ve bir aktivasyon fonksiyonu kullanarak birleştirir, ve bu, bir ağın karmaşık, doğrusal olmayan örüntüleri temsil edebilmesini sağlayan şeydir.
- Training, birçok örnek boyunca bir tahmin-ölç-hata-ayarla döngüsünü (forward pass, loss, backpropagation, gradient descent) tekrarlar -- derin ağları eğitmenin yavaş ve GPU-aç olmasının nedeni budur.

- Katmanlar, girdinin giderek daha soyut temsillerini öğrenme eğilimindedir; birçok katmanı yığmak bu kademeli soyutlamayı mümkün kılan şeydir.
- CNN, RNN ve transformer'lar, farklı veri türleri için üç yaygın mimaridir -- özellikle transformer'lar, bu kursta ilerleyen her büyük dil modelinin temelidir.

Cheat Sheet

- Neural network = input layer + hidden layer(lar) + output layer.
- Weight = iki node arasındaki bir bağlantı üzerindeki ayarlanabilir sayı.
- Activation function = bir node'un doğrusal olmayan örüntüleri temsil etmesini sağlayan şey.
- Loss function = bir tahminin ne kadar yanlış olduğunu ölçer.
- Backpropagation + gradient descent = ağırlıkların eğitim sırasında nasıl ayarlandığı.
- CNN = görüntüler. RNN = diziler (daha eski yaklaşım). Transformer = diziler (modern yaklaşım, LLM'lerin temeli).

Terimler Sözlüğü

- **Neural network (sinir ağı):** biyolojik nöronlardan gevşek biçimde ilham alan, basit bağlı birim ("nöron") katmanlarından oluşan bir öğrenme yapısı.
- **Layer (katman):** hepsi önceki katmandan girdi alan ve sonrakine çıktı ileten bir nöron grubu (girdi, gizli ya da çıktı).
- **Weight (ağırlık):** iki nöron arasındaki bir bağlantının gücünü temsil eden ayarlanabilir bir sayı.
- **Bias:** bir node'a ait, girdilerden bağımsız, ayarlanabilir sabit bir sayı -- weight'lerle birlikte bir ağın parametrelerini oluşturur.
- **Activation function (aktivasyon fonksiyonu):** bir node'un içinde uygulanan, bir ağın doğrusal olmayan örüntüleri temsil etmesini sağlayan bir fonksiyon.
- **Forward pass (ileri geçiş):** bir tahmin üretmek için bir örneği ağ boyunca girdiden çıktıya çalıştırmak.
- **Loss function (kayıp fonksiyonu):** bir tahminin doğru cevaptan ne kadar uzak olduğunu ölçer.
- **Backpropagation (geri yayılım):** bir ağ boyunca geriye doğru çalışarak her ağırlığın hataya katkısını belirleyen algoritma.
- **Gradient descent (gradyan inişi):** ağırlıkları kaybı azaltacak yönde itme süreci.
- **CNN / RNN / Transformer:** grid-benzeri veri, sıralı veri, ve (transformer durumunda) attention ile bir kerede işlenen sıralı veriye uygun üç neural network mimarisi.