5.Hafta

DERİN ÖĞRENME VE RNN MODELLERİ

1) Yapay Sinir Ağları (ANN) Mimarisi

Yapay Sinir Ağları (Artificial Neural Networks - ANN), insan beyninin çalışma prensibinden esinlenerek tasarlanmış hesaplamalı modellerdir. Temel olarak, birbirine bağlı **katmanlardan** ve **nöronlardan (birimlerden)** oluşurlar ve karmaşık kalıpları öğrenmek için verileri işlerler.

- **Nöron (Perceptron):** Bir sinir ağının temel işlem birimidir. Birden fazla girdi alır, her girdiyi bir ağırlıkla çarpar, tüm bunları toplar, bir sapma (bias) ekler ve son olarak bir **aktivasyon fonksiyonundan** geçirerek bir çıktı üretir.
- **Katmanlar:**
    - **Girdi Katmanı (Input Layer):** Modelin ilk katmanıdır ve ham veriyi (özellikleri) alır. Her bir özellik için bir nöron bulunur.
    - **Gizli Katmanlar (Hidden Layers):** Girdi ve çıktı katmanları arasındaki katmanlardır. Asıl karmaşık hesaplamaların ve özellik öğreniminin gerçekleştiği yerlerdir. Bir ağda birden fazla gizli katman bulunabilir. "Derin Öğrenme" terimi, birden fazla gizli katmana sahip ağları ifade eder.
    - **Çıktı Katmanı (Output Layer):** Modelin son katmanıdır ve modelin tahminini üretir. Sınıflandırma problemlerinde her sınıf için bir nöron olabilirken, regresyon problemlerinde tek bir nöron olabilir.
- **Ağırlıklar (Weights) ve Sapmalar (Biases):** Nöronlar arasındaki bağlantıların gücünü temsil eden sayılardır. Model, eğitim sırasında bu ağırlıkları ve sapmaları ayarlayarak verilerdeki kalıpları öğrenir.
- **Aktivasyon Fonksiyonları (Activation Functions):** Bir nöronun girdilerinin toplamını alıp, çıktıyı belirli bir aralığa (örneğin 0-1 veya -1-1) dönüştüren doğrusal olmayan fonksiyonlardır. Bu doğrusal olmama, ağın karmaşık ilişkileri öğrenmesini sağlar. Yaygın aktivasyon fonksiyonları: ReLU, Sigmoid, Tanh, Softmax.
- **İleri Besleme (Feedforward):** Girdinin ağdan tek yönlü olarak, girdi katmanından çıktı katmanına doğru ilerlemesi sürecidir.
- **Geriye Yayılım (Backpropagation):** Modelin tahmin hatalarını (maliyet fonksiyonu) kullanarak ağın ağırlıklarını ve sapmalarını güncellediği temel eğitim algoritmasıdır. Hata, çıktı katmanından girdi katmanına doğru geriye doğru yayılır ve her nöronun hataya katkısı hesaplanır.

2) Keras & TensorFlow ile Model Oluşturma

TensorFlow, Google tarafından geliştirilen, derin öğrenme ve sayısal hesaplama için kullanılan açık kaynaklı bir kütüphanedir.

Keras ise TensorFlow üzerinde çalışan yüksek seviyeli bir API'dir ve sinir ağlarını hızlı ve kolay bir şekilde oluşturmayı sağlar.

- **TensorFlow:** Düşük seviyeli operasyonlar, dağıtık eğitim, GPU desteği gibi karmaşık görevler için güçlüdür.
- **Keras:** Kullanım kolaylığı, hızlı prototipleme, modüler yapı sunar. Genellikle karmaşık detaylardan soyutlar ve yaygın derin öğrenme yapılarını önceden tanımlanmış katmanlar ve modellerle oluşturmayı sağlar.

3) RNN, LSTM, GRU Nedir?

Tekrarlayan Sinir Ağları (Recurrent Neural Networks - RNN), sıralı verilerle (metin, ses, zaman serileri) çalışmak için tasarlanmış özel bir ANN türüdür. Geleneksel ANN'lerin aksine, RNN'ler bir "belleğe" sahiptir; önceki zaman adımlarındaki bilgileri şimdiki zamana taşıyabilirler.

- **RNN Mantığı:** Bir RNN nöronu, bir girdiyi işlerken aynı zamanda önceki zaman adımında ürettiği çıktıyı da (veya gizli durumunu) girdi olarak kullanır. Bu "döngüsel" yapı, sıralı bağımlılıkları öğrenmesini sağlar.
- **Problemler:** Uzun dizilerde (**uzun vadeli bağımlılıklar**) bilgi aktarımında zorlanma (vanishing/exploding gradient problemi). Çok uzaktaki bilgiler, ağın derinliklerinde kaybolma eğilimindedir.

LSTM (Long Short-Term Memory)

LSTM'ler, RNN'lerin uzun vadeli bağımlılık sorununu çözmek için tasarlanmış özel bir RNN birimidir. Karmaşık bir "hücre durumu" ve bu durumu kontrol eden "geçit mekanizmaları" (girdi geçidi, unutma geçidi, çıktı geçidi) içerirler.

- **Geçitler:** Birimdeki bilginin ne kadarının tutulacağını, ne kadarının unutulacağını ve ne kadarının çıktı olarak iletileceğini düzenleyen kapılar gibi davranır. Bu sayede, LSTM'ler sıralı verilerde binlerce zaman adımı öncesindeki bilgileri bile hatırlayabilir.
- **Kullanım Alanları:** Metin çevirisi, konuşma tanıma, metin üretimi, duygu analizi.

GRU (Gated Recurrent Unit)

GRU'lar, LSTM'lere benzer ancak daha basitleştirilmiş bir yapıya sahip bir RNN birimidir. LSTM'deki üç geçit yerine sadece iki geçit (güncelleme geçidi ve reset geçidi) kullanır.

- **Avantajları:** LSTM'lere göre daha az parametreye sahip olduğu için genellikle daha hızlı eğitilir ve daha az hesaplama gücü gerektirir. Çoğu durumda LSTM'lere benzer performans gösterir.
- **Kullanım Alanları:** LSTM ile benzer senaryolarda kullanılabilir.

4) NLP ve Metin Verisi

Doğal Dil İşleme (Natural Language Processing - NLP), bilgisayarların insan dilini anlamasını, işlemesini ve üretmesini sağlayan yapay zeka alanıdır.

- **Metin Verisi:** Bilgisayarlar için ham metin verisi, doğrudan işlenebilir değildir. Sayısal formata dönüştürülmesi gerekir.

Tokenization, Padding, Embedding

Metin verilerini derin öğrenme modelleri için hazırlarken kullanılan temel ön işleme adımlarıdır:

- **Tokenization (Tokenleştirme):** Metni daha küçük birimlere (kelimeler, alt kelimeler veya karakterler) ayırma işlemidir. Bu küçük birimlere **token** denir.
    - **Örnek:** "Merhaba dünya!" -> ["Merhaba", "dünya", "!"]
- **Padding (Doldurma):** Derin öğrenme modelleri genellikle sabit uzunlukta girdiler bekler. Cümleler farklı uzunluklarda olduğundan, kısa cümleler belirli bir uzunluğa kadar sıfırlarla veya özel bir "padding" token'ıyla doldurulur (sağdan veya soldan), uzun cümleler ise kesilir.
- **Embedding (Gömme / Vektör Temsili):** Kelimeleri veya token'ları, anlamlarını ve anlamsal ilişkilerini koruyan yoğun, düşük boyutlu vektörlere dönüştürme işlemidir. Benzer anlama sahip kelimeler, vektör uzayında birbirine daha yakın konumlanır.

RNN ile Kelime Tahmini

RNN'ler, bir dizideki sonraki elemanı tahmin etmek için doğal olarak uygundur. **Kelime tahmini** (veya karakter tahmini), bir metin dizisi verildiğinde bir sonraki kelimeyi/karakteri tahmin etmeyi amaçlayan bir NLP görevidir.

- **Mantık:** Model, verilen kelimelerin dizisini alır, her kelimeyi bir gömme vektörüne dönüştürür ve RNN katmanları aracılığıyla bu diziyi işler. RNN'in gizli durumu, şimdiye kadar görülen dizinin bağlamını yakalar. Çıktı katmanı, sonraki kelimenin olasılık dağılımını tahmin eder.

Basit RNN/LSTM ile Kelime Tahmini Mimarisi:

1. **Metin Verisi:** "Merhaba dünya, nasılsın?"
2. **Tokenleştirme & Kelime-ID Eşlemesi:**
    - {"Merhaba": 1, "dünya": 2, "nasılsın": 3}
3. **Dizi Oluşturma (Girdi-Çıktı Çiftleri):**
    - Girdi: ["Merhaba"] -> Çıktı: ["dünya"]
    - Girdi: ["Merhaba", "dünya"] -> Çıktı: ["nasılsın"]
4. **Padding:** Tüm dizileri sabit bir uzunluğa getirme.
5. **Model:**
    - Embedding katmanı (kelimeleri vektörlere dönüştürür)
    - LSTM veya GRU katmanı (dizi bağımlılıklarını öğrenir)
    - Dense (Yoğun) katman (tüm kelime dağarcığı üzerinde bir sonraki kelime için olasılık dağılımı çıkarır)
    - Softmax aktivasyon (olasılıkları toplamı 1 yapacak şekilde normalize eder)

5) Encoder-Decoder Yapısı

Encoder-Decoder (Kodlayıcı-Çözücü) mimarisi, özellikle **diziden-diziye (sequence-to-sequence - Seq2Seq)** görevleri için kullanılır; yani bir giriş dizisini alıp farklı bir çıktı dizisi üreten görevler. Makine çevirisi, diyalog sistemleri ve metin özetleme gibi alanlarda yaygın olarak kullanılır.

- **Encoder (Kodlayıcı):**
    - Girdi dizisini (örneğin, kaynak dildeki cümle) alır.
    - Bu diziyi adım adım işler ve tüm dizinin bilgisini özetleyen sabit boyutlu bir **bağlam vektörüne (context vector)** sıkıştırır. Bu vektör, giriş dizisinin "düşüncesi" veya "anlamı" olarak düşünülebilir.
    - Genellikle bir RNN, LSTM veya GRU ağıdır.

- **Decoder (Çözücü):**
    - Encoder'dan gelen bağlam vektörünü girdi olarak alır.
    - Bu bağlam vektörünü kullanarak çıktı dizisini (örneğin, hedef dildeki cümle) adım adım üretir.
    - Her zaman adımında, bir önceki zaman adımında üretilen kelimeyi ve bağlam vektörünü kullanarak bir sonraki kelimeyi tahmin eder.
    - Genellikle bir RNN, LSTM veya GRU ağıdır.

- **Problemler:** Standart Encoder-Decoder yapısında, tüm giriş dizisi tek bir sabit boyutlu bağlam vektörüne sıkıştırılır. Uzun giriş dizileri için bu bağlam vektörü, tüm gerekli bilgiyi tutmakta zorlanabilir, bu da performans düşüşüne neden olur. Bu sorunu çözmek için **Attention mekanizması** geliştirilmiştir.

6) Attention Mekanizması

Attention (Dikkat) Mekanizması, Encoder-Decoder mimarisinin uzun dizilerle başa çıkma ve bilgiyi daha etkili kullanma yeteneğini artıran bir tekniktir. Encoder'ın tüm giriş dizisini tek bir sabit bağlam vektörüne sıkıştırması sorununu ortadan kaldırır.

- **Mantık:** Çözücü, çıktı dizisindeki her bir elemanı üretirken, giriş dizisinin **tüm elemanlarına doğrudan erişir** ve o anki çıktı için en alakalı olan giriş elemanlarına **"dikkat eder" (yani onlara daha fazla ağırlık verir)**.

- **Nasıl Çalışır?**
    - Encoder, her bir giriş zaman adımının gizli durumunu (temsilini) üretir.
    - Decoder, çıktı üretirken, her bir giriş gizli durumu ile kendi mevcut gizli durumu arasında bir "uyumluluk skoru" hesaplar.
    - Bu skorlar, giriş gizli durumları üzerinde bir **ağırlıklı ortalama** oluşturmak için kullanılır. Bu ağırlıklı ortalama, o anki çıktı için en alakalı bağlamı sağlar.
    - Bu "dikkatli" bağlam vektörü, her zaman adımında Decoder'a beslenir.

- **Avantajları:**
    - Uzun dizilerle çok daha iyi başa çıkar.
    - Modelin çıktı üretirken hangi giriş kısmına "baktığını" yorumlamak mümkün hale gelir.
    - Makine çevirisi ve karmaşık soru-cevap sistemlerinde devrim yaratmıştır.

7) Transformers Mimarisi

Transformers, 2017 yılında Google tarafından tanıtılan ve özellikle dikkat mekanizmasına dayanan, **RNN'lere ve CNN'lere (Evrişimli Sinir Ağları)** ihtiyaç duymayan devrim niteliğinde bir sinir ağı mimarisidir. NLP alanında (ve son zamanlarda bilgisayar görüşünde) hızla dominant hale gelmiştir.

- **Temel Fikir:** Tüm bir diziyi paralel olarak işler. RNN'lerin ardışık işleme zorunluluğunu ortadan kaldırarak hem eğitim hızını artırır hem de uzun vadeli bağımlılıkları çok daha etkili bir şekilde yakalar.
- **Ana Bileşenler:**
    - **Self-Attention (Kendi Kendine Dikkat):** Bir dizideki her bir elemanın, dizideki diğer tüm elemanlarla olan ilişkisini hesaplamasını sağlar. Bir kelimenin anlamını, cümledeki diğer kelimelerin bağlamına göre belirlemesine olanak tanır.
    - **Multi-Head Attention:** Dikkat mekanizmasının birden fazla kez (farklı "kafalar" kullanarak) paralel olarak çalıştırılması ve sonuçlarının birleştirilmesidir. Bu, modelin farklı türdeki ilişkileri aynı anda öğrenmesini sağlar.
    - **Konumsal Kodlamalar (Positional Encodings):** Transformers'lar ardışık bir yapıya sahip olmadığından, kelimelerin dizideki konum bilgilerini modele vermek için konumsal kodlamalar eklenir.
    - **Encoder ve Decoder Blokları:** Transformers mimarisi de Encoder-Decoder yapısını kullanır, ancak her blok içindeki katmanlar tamamen dikkat mekanizmalarına dayanır.
- **Avantajları:**
    - **Paralelleştirilebilirlik:** RNN'lerin aksine, tüm diziyi aynı anda işleyebilir, bu da GPU'lar üzerinde çok daha hızlı eğitim sağlar.
    - **Uzun Menzilli Bağımlılıklar:** Uzun dizilerdeki uzak kelimeler arasındaki ilişkileri mükemmel bir şekilde yakalar.
    - **Transfer Öğrenimi:** BERT, GPT (Generative Pre-trained Transformer) gibi büyük önceden eğitilmiş Transformer modelleri, birçok NLP görevi için transfer öğrenimi (fine-tuning) ile üstün sonuçlar verir.
- **Dezavantajları:**
    - Hesaplama ve bellek açısından pahalıdır (özellikle çok uzun diziler için).
    - RNN'lere göre daha fazla veri ve model parametresi gerektirebilir.

Kullanım Alanları (Sohbet Botu):

- **Niyet Tanıma:** Daha karmaşık ve nüanslı kullanıcı niyetlerini anlama.
- **Varlık Tanıma (Named Entity Recognition - NER):** Metinden kişi, yer, tarih gibi önemli bilgileri çıkarma.
- **Duygu Analizi:** Kullanıcı metninin duygu tonunu (pozitif, negatif, nötr) belirleme.
- **Metin Üretimi:** Daha akıcı ve bağlama uygun yanıtlar, özetler veya yaratıcı metinler üretme.
- **Makine Çevirisi:** Kullanıcı sorgularını farklı dillere çevirme.