VERİ BİLİMİ DERSİ

Derin Öğrenme ve Yapay Sinir Ağları

Hafta 10 · Modül 10
Nörondan Derin Ağlara: TensorFlow/Keras ile Uygulama

Dr. Murat Altun

Veri Bilimi ve Yapay Zekâ Eğitimi · 2026

6
Saat

3
Notebook

98%+
MNIST Hedef

İçindekiler

Biyolojik Nörondan Yapay Nörona

BÖLÜM 1

Biyolojik Nöron

1	**Dendrit**	Sinyalleri alır (girdiler)
2	**Soma (Hücre)**	Sinyalleri işler (toplama)
3	**Akson**	Çıktıyı iletir
4	**Sinaps**	Diğer nöronlara bağlantı (ağırlık)
5	**Eşik değeri**	Ateşleme kararı (aktivasyon)

→

Yapay Nöron (Perceptron)

1	**Girdiler (x)**	Veri özellikleri (features)
2	**Ağırlıklar (w)**	Bağlantı gücü (öğrenilir)
3	**Toplama (Σ)**	$w_1x_1 + w_2x_2 + \ldots + b$
4	**Aktivasyon (f)**	Doğrusal olmayan dönüşüm
5	**Çıktı (ŷ)**	Tahmin değeri

Formül: $\hat{y} = f(\Sigma\, w_i x_i + b)$ → *Biyolojik nöronun matematiksel modeli*

Yapay Sinir Ağı (ANN) Mimarisi

BÖLÜM 1

Input Layer (Giriş Katmanı)

Veri özelliklerini alır.
Her özellik bir nöron.
Örnek: 784 piksel → 784 nöron

Hidden Layer(s) (Gizli Katmanlar)

Öğrenmenin gerçekleştiği yer.
Her nöron bir özellik öğrenir.
Derin = çok gizli katman

Output Layer (Çıkış Katmanı)

Sonuç üretir.
Sınıflandırma: softmax
Regresyon: lineer

Ağırlıklar (Weights) ve Bias

- Ağırlık (w): Bağlantının önemi — büyük w = güçlü sinyal
- Bias (b): Eşik değeri — nöronun ne kadar kolay ateşleneceği
- Eğitim = En iyi w ve b değerlerini bulmak

Temel Terminoloji

Derin Öğrenme	2+ gizli katmanlı ağ
Fully Connected	Her nöron bir sonraki katmana bağlı
Parametre	Toplam ağırlık + bias sayısı

Aktivasyon Fonksiyonları

Fonksiyon	Formül	Aralık	Kullanım Alanı
ReLU	$\max(0, x)$	[0, +∞)	Gizli katmanlar (varsayılan)
Sigmoid	$1 / (1 + e^{-x})$	(0, 1)	İkili sınıflandırma çıkışı
Softmax	$e^{x_i} / \Sigma e^{x_j}$	(0, 1) toplam=1	Çoklu sınıf çıkışı
Tanh	$(e^{x} - e^{-x})/(e^{x} + e^{-x})$	(-1, 1)	RNN, gizli katmanlar
Leaky ReLU	$\max(0.01x, x)$	(-∞, +∞)	Dying ReLU sorununa çözüm

💡 **Pratik Kural: Gizli katman → ReLU | İkili çıkış → Sigmoid | Çoklu sınıf → Softmax**

İleri Yayılım (Forward Propagation)

BÖLÜM 1

1 Girdi Al

$x_1, x_2, \ldots, x_n$ özellik değerleri ağa verilir

2 Ağırlıkla Çarp

Her girdi kendi ağırlığıyla çarpılır: $w_i \times x_i$

3 Topla + Bias

$z = \Sigma(w_i \times x_i) + b$ (ağırlıklı toplam)

4 Aktivasyon Uygula

$a = f(z)$ — doğrusal olmayan dönüşüm

5 Sonraki Katmana

Çıktı bir sonraki katmanın girdisi olur

6 Tahmin Üret

Son katman çıktısı = modelin tahmini ($\hat{y}$)

```
Tam Akış:   x → (w·x + b) → f(z) → a → ... → ŷ    (katman katman ilerler)
```

Geri Yayılım (Backpropagation)

Gradient Descent (Gradyan İnişi)

Amaç: Loss fonksiyonunu minimize etmek.

1. Tahmin yap (forward pass)
2. Hatayı hesapla (loss)
3. Gradyanları hesapla ($\partial L/\partial w$)
4. Ağırlıkları güncelle:
   $w = w - \eta \times \partial L/\partial w$

Zincir Kuralı (Chain Rule)

Türev zinciri ile her ağırlığın hataya katkısı hesaplanır:

$\partial L/\partial w_1 = \partial L/\partial \hat{y} \times \partial \hat{y}/\partial z \times \partial z/\partial w_1$

- Çıkıştan girişe doğru geriye gider
- Her katmandaki gradyan hesaplanır
- Tüm ağırlıklar eş zamanlı güncellenir

Backpropagation Döngüsü

→

Forward Pass

Girdi → Tahmin

⚡

Loss Hesapla

Hata ölçümü

←

Backward Pass

Gradyan hesapla

↻

Güncelle

$w = w - \eta \nabla L$

Loss (Kayıp) Fonksiyonları

MSE (Mean Squared Error)

$$L = (1/n)\ \Sigma(y_i - \hat{y}_i)^2$$

Regresyon problemleri

Büyük hatalara daha fazla ceza verir.
Ev fiyat tahmini, sıcaklık tahmini.

Binary Cross-Entropy

$$L = -[y \cdot \log(\hat{y}) + (1-y) \cdot \log(1-\hat{y})]$$

İkili sınıflandırma

Sigmoid çıkışı ile kullanılır.
Spam/normal, hasta/sağlıklı.

Categorical Cross-Entropy

$$L = -\Sigma\ y_i \cdot \log(\hat{y}_i)$$

Çoklu sınıflandırma

Softmax çıkışı ile kullanılır.
MNIST (0–9), hayvan türü.

Optimizer'lar: SGD vs Adam

SGD (Stochastic Gradient Descent)

- En temel optimizer
- $w = w - \eta \times \nabla L$
- Sabit learning rate kullanır
- Basit ama yavaş yakınsama
- Momentum eklenebilir (hız artışı)
- Avantaj: Anlaması kolay, az bellek

Adam (Adaptive Moment)

- Momentum + RMSProp birleşimi
- Adaptif learning rate (her w için)
- Hızlı yakınsama
- Varsayılan tercih (çoğu projede)
- β_1=0.9, β_2=0.999, ε=1e-8
- Avantaj: Hiper parametre hassasiyeti az

Learning Rate (η) — Öğrenme Hızı

η çok küçük	Çok yavaş öğrenir, epoch israfı
η ideal	Hızlı ve kararlı yakınsama
η çok büyük	Salınım yapar, öğrenemez

TensorFlow ve Keras

BÖLÜM 3

TensorFlow Nedir?

- Google tarafından geliştirilen açık kaynak kütüphane
- Tensör tabanlı hesaplama grafikleri
- GPU/TPU desteği ile hızlı eğitim
- Üretim ortamı (TF Serving, TF Lite)
- Dünya çapında en yaygın DL framework'ü

Keras Neden?

- TensorFlow'un yüksek seviye API'si
- Hızlı prototipleme (az kodla çok iş)
- Okunabilir ve sezgisel sözdizimi
- Eğitim için ideal başlangıç noktası
- tf.keras olarak TF 2.x'e entegre

TensorFlow Ekosistemi

TensorFlow Core	Düşük seviye hesaplama	**Keras (tf.keras)**	Yüksek seviye model API	**TF Data**	Veri pipeline'ları
TensorBoard	Görselleştirme aracı	**TF Lite**	Mobil/Edge cihazlar	**TF Serving**	Üretim deploy

Keras Sequential Model Oluşturma


```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

# Model oluştur
model = Sequential([
    Dense(128, activation='relu', input_shape=(784,)),
    Dense(64, activation='relu'),
    Dense(10, activation='softmax')
])

# Compile
model.compile(
    optimizer='adam',
    loss=
'categorical_crossentropy',
    metrics=
['accuracy']
)
```

Katman 1 — Dense(128)

128 nöron, ReLU aktivasyon.
784 girdi → 128 çıktı.

Katman 2 — Dense(64)

64 nöron, ReLU aktivasyon.
Özellik çıkarma katmanı.

Katman 3 — Dense(10)

10 nöron, Softmax aktivasyon.
10 sınıf olasılıkları.

`model.summary()` → Toplam parametre: 109,386 (128×784 + 128 + 64×128 + 64 + 10×64 + 10)

Eğitim Hiper Parametreleri

Epoch

Tüm veriyi kaç kez göreceği

Batch

Bir adımda kaç örnek işleneceği

η (LR)

Ağırlık güncelleme adım büyüklüğü

Epoch Sayısı

- 1 epoch = tüm eğitim verisinin 1 tur geçişi
- Az epoch → underfitting (yetersiz öğrenme)
- Çok epoch → overfitting (ezber)
- Tipik: 10–100 epoch (veri boyutuna göre)
- Early Stopping ile otomatik durdurma

Batch Size

- Mini-batch: veriyi parçalar halinde işle
- Küçük batch (32): daha iyi genelleme, yavaş
- Büyük batch (256+): hızlı ama bellek gerekir
- Varsayılan: 32 (dengeli tercih)
- GPU belleğine göre ayarlanır

Learning Rate

- En kritik hiper parametre
- Tipik başlangıç: 0.001 (Adam için)
- LR Scheduler ile eğitim sırasında azaltma
- ReduceLROnPlateau: plato → küçült
- Warmup: düşük LR ile başla, artır

MNIST El Yazısı Rakam Tanıma

BÖLÜM 3

60K	10K	28×28	0–9
Eğitim Görüntüsü	Test Görüntüsü	Piksel Boyutu	Sınıf Sayısı

Veri Seti Hakkında

- Yann LeCun tarafından oluşturuldu (1998)
- "Hello World" of Deep Learning
- Gri tonlama: 0 (siyah) – 255 (beyaz)
- Normalize: piksel/255 → [0,1]
- Flatten: 28×28 → 784 boyutlu vektör
- Label: One-hot encoding (10 sınıf)
- Keras'ta hazır: tf.keras.datasets.mnist

Ön İşleme Adımları

1. Veriyi yükle: mnist.load_data()
2. Normalize et: X / 255.0
3. Flatten: reshape(-1, 784)
4. One-hot: to_categorical(y, 10)
5. Train/Test split (hazır geliyor)

MNIST — Tam Kod Örneği


```
import tensorflow as tf
from tensorflow.keras.utils import to_categorical

# 1. Veri yükle
(X_train, y_train), (X_test, y_test) = \
    tf.keras.datasets.mnist.load_data()

# 2. Ön işleme
X_train = X_train.reshape(-1, 784) / 255.0
X_test  = X_test.reshape(-1, 784) / 255.0
y_train = to_categorical(y_train, 10)
y_test  = to_categorical(y_test, 10)

# 3. Model
model = tf.keras.Sequential([
    Dense(256, activation='relu', input_shape=(784,)),
    Dense(128, activation='relu'),
    Dense(10,  activation='softmax')
])

# 4. Eğit
model.compile(optimizer='adam', loss='categorical_crossentropy',
                metrics=[
'accuracy'])
model.fit(X_train, y_train, epochs=20, batch_size=32,
          validation_split=
0.2)

# 5. Değerlendir
loss, acc = model.evaluate(X_test, y_test)
print(f'Test Accuracy: {acc:.4f}')
```

Beklenen Sonuçlar

- Eğitim acc: ~99.5%
- Test acc: ~98.2%
- Eğitim süresi: ~2 dk
- Parametre: ~235K

Önemli Notlar

- validation_split ile aşırı öğrenme takibi
- Epoch artışı = daha iyi (bir yere kadar)
- batch_size GPU belleğine göre ayarla

Overfitting (Aşırı Öğrenme) Önleme

🎲 Dropout

Eğitim sırasında rastgele nöronları kapat.
Her epoch'ta farklı alt ağ eğitilir.
Model tek bir yola bağımlı kalmaz.

```
model.add(Dropout(0.3))
# %30 nöron rastgele kapatılır
```

⏱ Early Stopping

Validation loss artmaya başlayınca dur.
Patience: kaç epoch bekle.
En iyi modeli geri yükle.

```
EarlyStopping(
  patience=5,
  restore_best_weights=True
)
```

⚖ Regularization

Ağırlıklara ceza ekle (L1/L2).
Büyük ağırlıkları küçültür.
Model basitliğini teşvik eder.

```
Dense(64,
  kernel_regularizer=
    l2(0.01)
)
```

Google Quick, Draw! ile Çizim Tanıma

BÖLÜM 3

Quick, Draw! Veri Seti

- Google'ın interaktif çizim oyunu verileri
- 345 kategori, 50M+ çizim
- Her çizim: 28×28 piksel (MNIST benzeri)
- Gerçek insan çizimleri (gürültülü veri)
- .npy formatında indirilebilir
- quickdraw.readthedocs.io

345
Kategori

50M+
Çizim

28px
Boyut

.npy
Format

Notebook Uygulaması: 10 Kategori Sınıflandırma

1. 10 kategori seç (kedi, köpek, araba, ev, ağaç...)
2. Her kategoriden 10.000 çizim indir (.npy)
3. Normalize et (/ 255.0) ve train/test split
4. Dense ANN modeli oluştur ve eğit
5. Confusion matrix ile sonuçları değerlendir

ANN vs Klasik Makine Öğrenmesi

BÖLÜM 4

Kriter	Klasik ML (RF, SVM, XGB)	Yapay Sinir Ağları (ANN)
📊 Doğruluk (Tabular)	Genellikle daha iyi (XGBoost güçlü)	Tabular veride bazen geride
🖼️ Doğruluk (Görüntü)	Sınırlı (özellik çıkarma gerekir)	Çok üstün (CNN ile)
⏱️ Eğitim Süresi	Hızlı (dakikalar)	Yavaş (GPU gerekebilir)
📦 Veri Gereksinimi	Az veriyle çalışabilir	Çok veri ister (1000+)
🔍 Yorumlanabilirlik	Yüksek (feature importance)	Düşük (kara kutu)
⚙️ Özellik Mühendisliği	Manuel gerekir	Otomatik öğrenir

Sonuç: Tabular veri → Klasik ML (XGBoost) | Görüntü/Metin/Ses → Derin Öğrenme (ANN/CNN/RNN)

Haftalık Notebook'lar

🧠 ann_temel.ipynb

Yapay sinir ağı temellerini öğrenme notebook'u.

1. Basit ANN oluşturma (from scratch)
2. Aktivasyon fonksiyonlarını görselleştirme
3. Forward/Backward pass implementasyonu
4. Numpy ile gradient descent

✏️ quickdraw.ipynb

Google Quick, Draw! çizim tanıma uygulaması.

1. 10 kategori veri indirme (.npy)
2. Keras Sequential model oluşturma
3. Eğitim ve validation takibi
4. Confusion matrix analizi

📈 ann_regresyon.ipynb

ANN ile regresyon problemi çözme.

1. Boston/California Housing veri seti
2. MSE loss ile model eğitimi
3. Tahmin vs gerçek scatter plot
4. Klasik ML ile karşılaştırma

Ödev ve Kaynaklar

Bu Hafta Yapılacaklar

1. **MNIST %98+ doğruluk hedefi**
   Katman sayısı, nöron sayısı, dropout ile deneme
2. **QuickDraw 10 kategori sınıflandırma**
   Kendi seçtiğin 10 kategori ile model eğit
3. **ANN Regresyon notebook**
   Housing veri seti ile fiyat tahmini
4. **Hiperparametre deneyleri**
   Epoch, batch, LR değiştirerek sonuçları karşılaştır
5. **TensorBoard görselleştirme**
   Eğitim/validation loss grafiklerini incele

Önerilen Kaynaklar

- **TensorFlow Resmi Dökümantasyon**
  tensorflow.org/tutorials
- **3Blue1Brown — Neural Networks**
  youtube.com (görsel açıklama)
- **Deep Learning with Python**
  François Chollet (Keras yaratıcısı)
- **Google Quick, Draw! Dataset**
  quickdraw.withgoogle.com
- **Keras Örnekleri**
  keras.io/examples
- **Playground TensorFlow**
  playground.tensorflow.org

Hafta 10 — Özet

1. Yapay nöron, biyolojik nöronun matematiksel modelidir: $\hat{y} = f(\Sigma wx + b)$
2. Forward → Loss → Backward → Update döngüsü ile ağ öğrenir
3. Keras ile 10 satır kodda güçlü modeller oluşturulabilir
4. Dropout + Early Stopping + Regularization ile overfitting önlenir
5. Bölüm 3 başlıyor: Derin öğrenme yolculuğu artık CNN ve RNN ile devam edecek

"Yapay sinir ağları, insan beyninin öğrenme biçimini taklit eden en güçlü araçtır."

Dr. Murat Altun · Veri Bilimi ve Yapay Zekâ Eğitimi · 2026