BCDataWorks

YAPAY SİNİR AĞLARINA GİRİŞ

Perceptron ve Multilayer Perceptron

Temel matematikten backpropagation'a, kredi riski uygulamasından Python örneklerine kadar tek bir çalışma notu.

· Kavramsal anlatım · Görsel şemalar · Uygulamalı örnekler

Burak Ceylan · bcdataworks.com · Çalışma Notları № 01

Bu doküman ne anlatıyor?

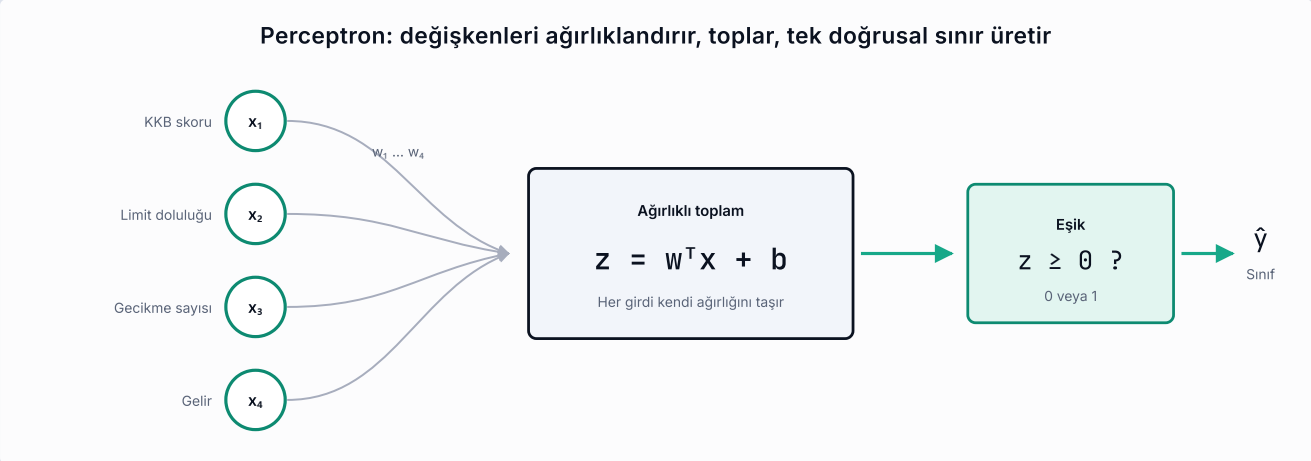
Bu çalışma, yapay sinir ağlarının en temel yapı taşı olan perceptrondan başlayarak Multilayer Perceptron (MLP) mimarisine kadar ilerler. Amaç yalnızca formülleri vermek değil; modelin neyi, neden ve nasıl öğrendiğini sezgisel biçimde kurmaktır.

> **Bir cümlede fark**
> Perceptron tek bir doğrusal karar sınırı öğrenir; MLP ise çok sayıda nöronu ve doğrusal olmayan aktivasyonları birleştirerek karmaşık karar bölgeleri oluşturur.

İçindekiler

01 Perceptron: en sade yapay nöron

Perceptron, bir gözlemi iki sınıftan birine ayırmak için geliştirilmiş en temel yapay sinir ağı modelidir. Bir müşteriyi “iyi” veya “riskli”, bir işlemi “normal” veya “şüpheli” olarak sınıflandırmak gibi ikili problemlerde kullanılabilir.



Şekil 1. Perceptronun girdi, ağırlıklı toplam ve eşik adımları.

$$z = w_1x_1 + w_2x_2 + \ldots + w_nx_n + b = w^Tx + b$$

x: girdiler, w: ağırlıklar, b: bias, z: doğrusal skor

Model önce her değişkeni ilgili ağırlığıyla çarpar, bu değerleri toplar ve bias ekler. Ardından z skoru bir eşik fonksiyonundan geçirilir. Sonuç pozitif taraftaysa sınıf 1, negatif taraftaysa sınıf 0 seçilir.

$$\hat{y} = 1 \ (z \geq 0) \ | \ \hat{y} = 0 \ (z < 0)$$

Klasik perceptron çıktısı olasılık değil, doğrudan sınıftır.

02 Sezgisel kredi riski örneği

İki değişken kullanan basit bir model düşünelim: limit doluluk oranı (x_1) ve son dönem gecikme sayısı (x_2). Model aşağıdaki skoru öğrenmiş olsun:

$$z = 2x_1 + 0{,}8x_2 - 3$$

Limit doluluğu %90 ve gecikme sayısı 3 olan müşteri için z = 2×0,90 + 0,8×3 − 3 = 1,2 bulunur. Skor sıfırın üstünde olduğundan perceptron müşteriyi **riskli** sınıfa atar.

Ağırlık neyi ifade eder?

Pozitif ağırlık, değişken yükseldiğinde sınıf 1 yönündeki skoru artırır; negatif ağırlık azaltır. Ağırlığın büyüklüğü etkinin gücü hakkında fikir verir; ancak değişken ölçekleri farklıysa doğrudan karşılaştırma yanıltıcı olabilir.

03 Perceptron nasıl öğrenir?

Model başlangıçta sıfır veya küçük rastgele ağırlıklarla başlar. Her gözlemde tahmin yapar, gerçek sınıfla farkı hesaplar ve **yalnızca hata varsa** ağırlıkları düzeltir.

$$e = y - \hat{y}$$

$$w_i(\text{yeni}) = w_i(\text{eski}) + \eta \ (y - \hat{y}) \ x_i$$

$$b(\text{yeni}) = b(\text{eski}) + \eta \ (y - \hat{y})$$

η: learning rate / öğrenme oranı

- Tahmin doğruysa y − ŷ = 0 olur ve ağırlıklar değişmez.
- Gerçek sınıf 1, tahmin 0 ise ağırlıklar gözlemi pozitif tarafa taşıyacak yönde güncellenir.
- Gerçek sınıf 0, tahmin 1 ise güncelleme ters yönde yapılır.
- Verinin tamamından bir kez geçilmesine **epoch** denir.

Yakınsama koşulu

Klasik perceptron algoritması, sınıflar doğrusal olarak ayrılabiliyorsa sonlu sayıda adımda ayıran bir sınır bulabilir. Sınıflar iç içeyse hatasız çözüme ulaşması beklenmez.

04 Karar sınırı ve XOR problemi

Perceptronun karar sınırı $w^Tx + b = 0$ eşitliğidir. İki değişkende bu sınır bir doğru, üç değişkende düzlem, daha yüksek boyutta hiperdüzlem olarak düşünülür.

Şekil 2. Solda tek doğruyla ayrılabilen sınıflar; sağda XOR düzeni.

XOR probleminde aynı sınıfa ait noktalar çapraz konumdadır. Tek bir doğru, iki sınıfı hatasız biçimde ayıramaz. Bu basit örnek, gizli katmanların neden gerekli olduğunu gösterir: birden fazla sınır üretip bunları birleştirmek gerekir.

05 Perceptron ve lojistik regresyon

İki model de önce $z = w^T x + b$ doğrusal skorunu hesaplar. Ayrım, bu skoru çıktıya dönüştürme biçimindedir.

- Perceptron keskin eşik kullanır ve doğrudan 0/1 sınıfı üretir.
- Lojistik regresyon sigmoid fonksiyonu kullanır ve 0 ile 1 arasında olasılık üretir.
- Lojistik regresyonda cut-off değiştirilebilir, kalibrasyon incelenebilir ve katsayılar daha rahat yorumlanabilir.
- Bu nedenle klasik kredi skorlama problemlerinde perceptrondan çok lojistik regresyon tercih edilir.

$$P(y=1 \mid x) = 1 / (1 + e^{-z})$$

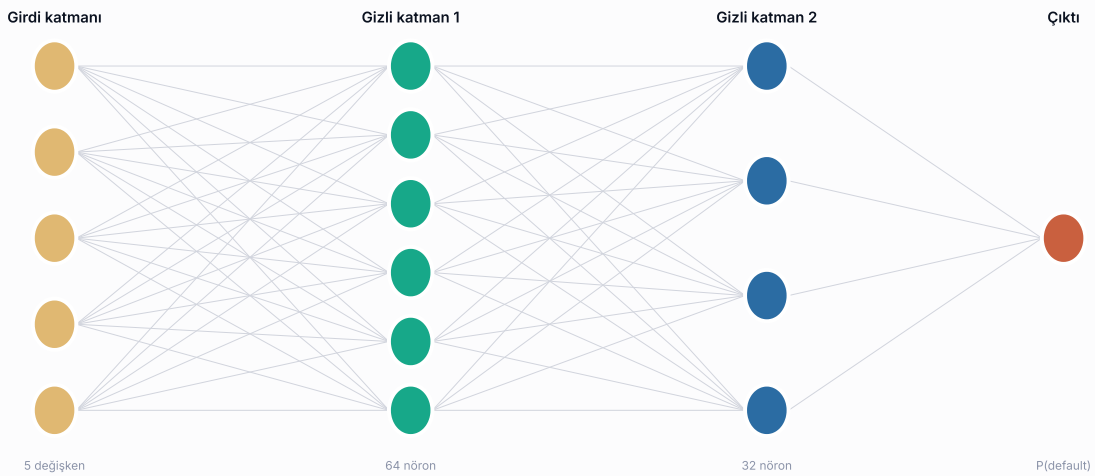
Lojistik regresyonun sigmoid çıktısı

Pratikte ne anlama geliyor?

Kredi kararında ihtiyaç duyulan şey çoğu zaman “riskli mi, değil mi” etiketi değil, sıralanabilir ve eşiklenebilir bir olasılıktır. Perceptronun keskin eşiği bu bilgiyi daha kaynağında atar; bu yüzden skorkart dünyasında karşılığı sınırlıdır.

06 Multilayer Perceptron (MLP)

MLP, birden fazla nöronun katmanlar halinde birbirine bağlandığı ileri beslemeli bir yapay sinir ağıdır. Temel yapı girdi katmanı, bir veya daha fazla gizli katman ve çıktı katmanından oluşur.



Her gizli nöron farklı bir özellik bileşimi öğrenir; sonraki katmanlar bunları bir araya getirir

Şekil 3. 5 → 6 → 4 → 1 biçimindeki örnek MLP mimarisi.

> **MLP'nin temel gücü**
> Her gizli nöron, girdilerin farklı ağırlıklı birleşimini öğrenir. Sonraki katmanlar bu ara temsilleri birleştirerek tek perceptronun kuramayacağı karmaşık karar bölgeleri üretir.

07 Katmanlar ne yapar?

Girdi katmanı

Her düğüm bir değişkeni temsil eder. Örneğin KKB skoru, limit doluluğu, gecikme sayısı, gelir ve borç/gelir oranı beş giriş düğümü olabilir. Girdi katmanı çoğunlukla hesaplama yapmaz; değerleri ilk gizli katmana taşır.

Gizli katman

Her gizli nöron gelen değerlerin ağırlıklı toplamını hesaplar ve sonucu bir aktivasyon fonksiyonundan geçirir:

$$z_j = \Sigma_i \, w_{ji} x_i + b_j \rightarrow a_j = f(z_j)$$

Farklı nöronlar farklı ağırlıklar öğrendiği için "yüksek limit doluluğu + düşük KKB skoru", "nakit çekim artışı + ödeme davranışındaki bozulma" gibi örtük kombinasyonları temsil edebilir.

Çıktı katmanı

İkili sınıflandırmada genellikle tek çıktı nöronu ve sigmoid kullanılır. Çok sınıflı problemlerde sınıf sayısı kadar çıktı nöronu ve softmax; regresyonda ise çoğunlukla doğrusal çıktı tercih edilir.

08 Aktivasyon fonksiyonu neden gerekli?

Katmanlar yalnızca doğrusal dönüşümlerden oluşsaydı, ne kadar çok katman eklenirse eklensin bütün ağ tek bir doğrusal dönüşüme indirgenebilirdi. Aktivasyon fonksiyonları araya **doğrusal olmayanlık** ekleyerek MLP'ye eğrisel sınırlar, eşik etkileri ve değişken etkileşimleri öğrenme gücü kazandırır.

Şekil 4. Yaygın aktivasyon fonksiyonlarının biçimi.

Sigmoid

Çıktıyı 0 ile 1 arasına sıkıştırır. İkili sınıflandırmanın çıktı katmanında uygundur; fakat uç değerlerde türevi çok küçüldüğü için gizli katmanlarda **vanishing gradient** sorununa yol açabilir.

Tanh

Çıktısı −1 ile 1 arasındadır ve sıfır merkezlidir. Sigmoide göre bazı avantajları olsa da uç bölgelerde gradyan küçülmesi yaşayabilir.

ReLU

ReLU(z) = max(0, z) biçimindedir. Pozitif bölgede basit ve güçlü gradyan sağladığı için gizli katmanlarda yaygın kullanılır. Sürekli negatif bölgede kalan nöronların öğrenememesi **"dying ReLU"** olarak bilinir.

Softmax

Birden fazla sınıf için her sınıfa toplamı 1 olan olasılıklar üretir. Düşük, orta ve yüksek risk gibi çok sınıflı çıkışlarda kullanılabilir.

09 İleri yayılım (forward propagation)

İleri yayılım, girdilerin katmanlardan sırasıyla geçirilerek tahminin hesaplanmasıdır. İki gizli katmanlı bir ağda temel akış şöyledir:

$$a^{(1)} = f(W^{(1)}x + b^{(1)})$$

$$a^{(2)} = f(W^{(2)}a^{(1)} + b^{(2)})$$

$$\hat{y} = \text{sigmoid}(W^{(3)}a^{(2)} + b^{(3)})$$

Örneğin ŷ = 0,78 çıkması, modelin müşteriyi riskli sınıfa güçlü biçimde yakın gördüğünü gösterir. Ancak bunun doğrudan %78 PD olarak yorumlanabilmesi için **olasılık kalibrasyonu** ayrıca doğrulanmalıdır.

10 Kayıp fonksiyonu

Kayıp fonksiyonu tahminin ne kadar hatalı olduğunu sayısallaştırır. İkili sınıflandırmada en yaygın tercih **binary cross-entropy**'dir:

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

Gerçek sınıf 1 olduğunda modelin 1'e yakın olasılık vermesi, gerçek sınıf 0 olduğunda 0'a yakın olasılık vermesi ödüllendirilir. Yanlış ve aşırı emin tahminlerin cezası büyüktür.

11 Backpropagation: hata geriye nasıl taşınır?

Şekil 5. Bir eğitim adımındaki ileri ve geri akış.

Backpropagation, **zincir kuralını** kullanarak kaybın her ağırlığa göre türevini hesaplar. Böylece ağ, tahmin hatasının hangi bağlantılardan ne ölçüde kaynaklandığını belirler.

$$w(yeni) = w(eski) - \eta \cdot \partial L/\partial w$$

Kavramsal ayrım

Backpropagation gradyanları **hesaplayan** yöntemdir. Gradient descent, Adam veya benzeri optimizasyon algoritmaları ise bu gradyanları **kullanarak** ağırlıkları günceller.

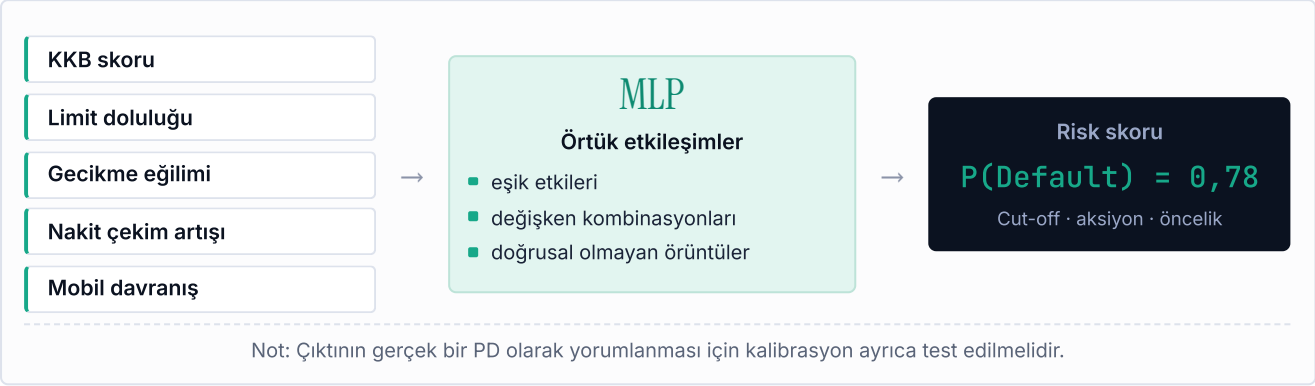
12 Gradient descent, batch ve epoch

- **Batch gradient descent:** tüm veri kullanıldıktan sonra tek güncelleme yapar; kararlı fakat maliyetlidir.
- **Stochastic gradient descent:** her gözlemden sonra günceller; hızlı fakat gürültülüdür.
- **Mini-batch gradient descent:** veriyi 32, 64, 128 veya 256 gibi gruplara böler; modern uygulamalarda en yaygın yaklaşımdır.
- **Epoch:** modelin eğitim verisinin tamamını bir kez görmesidir.
- **Learning rate:** ağırlıkların her adımda ne kadar değişeceğini belirler.

100.000 gözlem ve 1.000 batch size için bir epoch içinde 100 güncelleme yapılır. Eğitim 50 epoch sürerse yaklaşık 5.000 güncelleme gerçekleşir.

13 MLP neden doğrusal olmayan ilişkileri öğrenir?

Tek perceptron tek bir hiperdüzlem oluşturur. Gizli katmandaki çok sayıda nöron farklı doğrusal sınırlar üretir; sonraki katmanlar bu sınırların oluşturduğu bölgeleri birleştirir. Böylece MLP, tek bir değişkenin etkisinin diğer değişkenlerin seviyesine göre değiştiği **etkileşimleri** öğrenebilir.



Şekil 6. Kredi riski değişkenlerinden MLP risk skoruna örnek akış.

Örneğin limit doluluğu tek başına her müşteri için aynı risk etkisine sahip olmayabilir. Düşük KKB skoru, artan nakit çekim ve son ödeme davranışındaki bozulmayla birlikte etkisi çok daha kuvvetli hale gelebilir. MLP bu tür ilişkileri açık bir etkileşim değişkeni yazmadan öğrenebilir.

14 Perceptron ve MLP karşılaştırması

Özellik	Perceptron	MLP
Yapı	Tek hesaplama birimi	Birden fazla katman ve nöron
Karar sınırı	Doğrusal	Doğrusal olmayan
XOR problemi	Çözemez	Çözebilir
Aktivasyon	Keskin eşik	ReLU, tanh, sigmoid vb.
Öğrenme	Perceptron güncellemesi	Backpropagation + optimizasyon
Yorumlanabilirlik	Görece yüksek	Daha düşük
Overfitting riski	Düşük	Mimariye bağlı olarak yüksek
Tipik kullanım	Temel doğrusal sınıflandırma	Karmaşık örüntü ve etkileşimler

15 MLP mimarisi nasıl seçilir?

Katman ve nöron sayısı için her probleme uyan tek bir formül yoktur. Daha büyük ağ daha güçlü görünse de gereksiz parametre, overfitting ve eğitim maliyeti yaratabilir.

- Başlangıç noktası olarak tek veya iki gizli katman denenebilir: (32), (64, 32) veya (128, 64, 32).
- Nöron sayısı çok azsa model underfitting yaşayabilir; çok fazlaysa validation performansı bozulabilir.
- Mimari seçiminde yalnızca train skoru değil, validation ve out-of-time sonuçları kullanılmalıdır.
- Learning rate ve regularization çoğu zaman katman sayısı kadar belirleyicidir; early stopping hem eğitim süresini hem overfitting riskini azaltır.

16 Overfitting nasıl önlenir?

- Train–validation–test ayrımı kurun; kredi riskinde zaman bazlı **OOT** testi tercih edin.
- Validation loss iyileşmediğinde **early stopping** uygulayın.
- **L2 regularization** ile çok büyük ağırlıkları cezalandırın.
- **Dropout** ile eğitim sırasında bazı nöronları geçici olarak kapatın.
- Gereksiz katman ve nöronları azaltın.
- Özellik mühendisliğini ve veri kalitesini mimari büyüklüğünden önce iyileştirin.

$$J(\text{yeni}) = J + \lambda \sum w^2$$

L2 regularization: büyük ağırlıklara ek ceza

17 Ölçeklendirme neden kritiktir?

Gelir 100.000 seviyesinde, limit doluluğu 0,85 seviyesinde olduğunda gradyanların dengesi bozulabilir ve optimizasyon zorlaşabilir. Bu nedenle MLP'den önce StandardScaler benzeri dönüşümler yaygın biçimde kullanılır.

$$x_{standard} = (x - \mu) / \sigma$$

Ağaç modellerinden fark

Karar ağaçları ve boosting modelleri çoğunlukla ölçeklendirmeye ihtiyaç duymaz. MLP, lojistik regresyon, SVM ve mesafe tabanlı modellerde ise ölçek belirleyici olabilir.

18 Sınıf dengesizliği

Default oranı %2 olan bir portföyde tüm müşterilere "good" diyen model %98 accuracy elde eder; fakat risk yönetimi açısından işe yaramaz. Bu yüzden **accuracy tek başına kullanılmamalıdır.**

- Class weight veya sample weight ile bad gözlemlere daha yüksek önem verilebilir.
- Undersampling / oversampling uygulanabilir; ancak validation ve OOT setlerinin doğal dağılımı korunmalıdır.
- Karar eşiği iş maliyetine göre optimize edilebilir.
- Focal loss gibi yöntemler zor örneklere daha fazla ağırlık verebilir.

Dengesiz portföyde hangi metrik?

Accuracy yerine, sıralama gücünü ve nadir sınıftaki kaliteyi birlikte gösteren bir metrik kümesi kullanılır:

Metrik	Neyi gösterir?	Neden önemlidir?
ROC-AUC / Gini	Sıralama gücü	Kredi riskinde temel ayrıştırma ölçütü
PR-AUC	Nadir pozitif sınıftaki kalite	Düşük default oranlarında daha duyarlı
KS	Good ve bad dağılımları arasındaki maksimum fark	Cut-off ve ayrıştırma değerlendirmesinde yaygın
Recall / Capture	Bad müşterilerin ne kadarının yakalandığı	Erken aksiyon kapasitesini gösterir
Precision	Riskli işaretlenenlerin ne kadarının gerçekten bad olduğu	Operasyonel maliyeti etkiler
Kalibrasyon	Tahmin edilen olasılığın gerçekleşme oranıyla uyumu	Skor PD olarak kullanılacaksa kritiktir
OOT stabilite	Zaman dışındaki performans	Gerçek üretim dayanıklılığını ölçer

19 Python ile Perceptron

Scikit-learn `Perceptron` sınıfı doğrudan olasılık üretmez. Bu nedenle ROC-AUC ve Gini için `predict_proba` yerine `decision_function` skoru kullanılabilir.

```
from sklearn.linear_model import Perceptron
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, roc_auc_score

model = Pipeline([
    ("scaler", StandardScaler()),
    ("perceptron", Perceptron(
        max_iter=1000,
        eta0=0.01,
        learning_rate="constant",
        class_weight="balanced",
        random_state=42
    ))
])

model.fit(X_train, y_train)
y_pred  = model.predict(X_test)
y_score = model.decision_function(X_test)   # olasılık değil, skor

auc  = roc_auc_score(y_test, y_score)
gini = 2 * auc - 1

print(classification_report(y_test, y_pred))
print("AUC :", auc)
print("Gini:", gini)
```

20 Python ile MLP

```
from sklearn.neural_network import MLPClassifier
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import (
    roc_auc_score,
    average_precision_score,
    classification_report
)

mlp = Pipeline([
    ("scaler", StandardScaler()),
    ("model", MLPClassifier(
        hidden_layer_sizes=(64, 32),
        activation="relu",
        solver="adam",
        learning_rate_init=0.001,
        batch_size=256,
        alpha=0.0001,              # L2
        max_iter=300,
        early_stopping=True,
        validation_fraction=0.15,
        n_iter_no_change=15,
        random_state=42
    ))
])

mlp.fit(X_train, y_train)
y_prob = mlp.predict_proba(X_test)[:, 1]
y_pred = (y_prob ⩾ 0.50).astype(int)

auc    = roc_auc_score(y_test, y_prob)
gini   = 2 * auc - 1
pr_auc = average_precision_score(y_test, y_prob)

print("ROC-AUC:", auc)
print("Gini   :", gini)
print("PR-AUC :", pr_auc)
print(classification_report(y_test, y_pred))
```

Mimari gösterimi

`hidden_layer_sizes=(64, 32)`, girdi ile çıktı arasında sırasıyla 64 ve 32 nöronlu iki gizli katman bulunduğu anlamına gelir.

21 Kredi riskinde MLP'nin yeri

MLP güçlü bir genel amaçlı modeldir; ancak klasik tabular kredi riski verilerinde her zaman en iyi seçenek değildir. Lojistik regresyon ve gradient boosting modelleri daha az ayarla güçlü sonuç verebilir ve açıklanabilirlik açısından avantajlıdır.

MLP daha anlamlı olabilir

- Çok büyük müşteri ve işlem hacmi varsa
- Yüzlerce veya binlerce değişken bulunuyorsa
- Mobil davranış, işlem dizileri veya zaman pencereleri modelleniyorsa
- Metin, çağrı merkezi, chat veya embedding gibi farklı veri türleri birleştiriliyorsa
- Örtük ve yüksek dereceli etkileşimlerin güçlü olduğu düşünülüyorsa

Mutlaka karşılaştırılması gereken modeller

- Lojistik regresyon / scorecard
- XGBoost, LightGBM ve CatBoost
- Random Forest
- Explainable Boosting Machine
- Basit benchmark ve maliyet bazlı karar kuralı

> **Model seçiminin doğru sorusu**
> "En yüksek Gini hangi modelde?" tek başına yeterli değildir. OOT performansı, kalibrasyon, stabilite, açıklanabilirlik, çalışma süresi, üretim maliyeti ve aksiyona dönüşebilirlik birlikte değerlendirilmelidir.

22 Son zihinsel model

$$\text{Perceptron: } \hat{y} = \text{step}(Wx + b)$$

$$\text{MLP: } h = f(W_1 x + b_1) \rightarrow \hat{y} = g(W_2 h + b_2)$$

Perceptron şu soruyu sorar: "Bu gözlemleri tek bir ağırlıklı toplam ve doğrusal sınırla ayırabilir miyim?" MLP ise "Girdilerin farklı kombinasyonlarını ara katmanlarda öğrenip bunları birleştirerek daha karmaşık bir karar fonksiyonu kurabilir miyim?" sorusunu çözer.

> **Özet**
> Perceptron yapay sinir ağlarının en sade doğrusal yapı taşıdır. MLP; gizli katmanlar, aktivasyon fonksiyonları, loss ve backpropagation sayesinde doğrusal olmayan ilişkileri öğrenir. Tabular kredi riskinde güçlü bir aday olsa da boosting ve lojistik regresyonla OOT, kalibrasyon ve açıklanabilirlik boyutlarında karşılaştırılmalıdır.

— ÇALIŞMA SONU —