AdaBoost ML Algoritması

Atıl Samancıoğlu

1 Giriş

AdaBoost (*Adaptive Boosting*), zayıf öğrenicileri (*weak learners*) bir araya getirerek güçlü bir model (*strong learner*) oluşturmayı amaçlayan bir **boosting** algoritmasıdır. AdaBoost, genellikle karar ağaçlarının *decision stump* (derinliği 1 olan karar ağaçları) versiyonlarını kullanır ve bu modellerin hatalarına göre ağırlıklandırma yapar.

2 Temel Prensipler

AdaBoost şu adımları takip eder:

1. Verinin tüm noktalarına eşit ağırlıkla başlanır.
2. İlk zayıf model (örneğin karar ağacı stump) eğitilir.
3. Yanlış sınıflandırılan örneklerin ağırlıkları artırılır, doğru sınıflandırılanların ağırlıkları azaltılır.
4. Ağırlıklar normalleştirilir ve yeni örnek seçimi için *bin* sistemi kurulur.
5. Süreç belirlenen sayıda iterasyon boyunca tekrar eder.

3 AdaBoost Fonksiyonu

AdaBoost'un çıktısı aşağıdaki gibi hesaplanır:

$$F(x) = \sum_{m=1}^{M} \alpha_m h_m(x)$$

Burada:

- $h_m(x)$: m. zayıf öğrenici,
- α_m: h_m için ağırlık,
- M: toplam zayıf model sayısıdır.

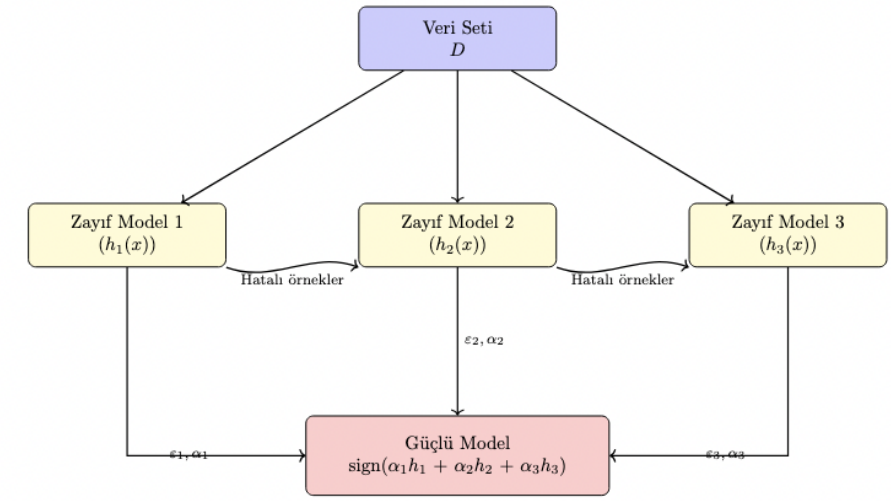

Figure 1: Adaboost Sürecinin Genel Şeması

4 AdaBoost Şeması

5 Örnek Veri Seti

Aşağıda 7 satırlık basit bir veri seti gösterilmektedir. Bu veri setinde iki özellik (**GPA, Interview Score**) ve bir hedef değişken (**Approval**) bulunmaktadır:

ID	**GPA**	**Interview Score**	**Approval (Y)**
1	<3.0	Düşük	No
2	<3.0	Yüksek	Yes
3	<3.0	Yüksek	Yes
4	>3.0	Düşük	No
5	>3.0	Yüksek	Yes
6	>3.0	Normal	Yes
7	<3.0	Normal	No

Table 1: AdaBoost için örnek veri seti

6 Adım 1: İlk Stump ve Tahminler

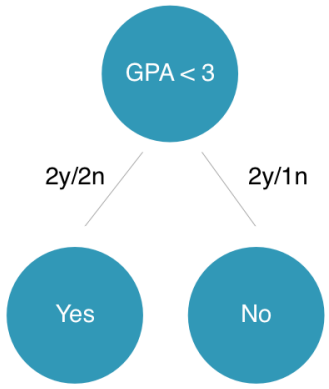

Figure 2: GPA Decision Tree Stump Adayı

Seçilen stump: *Interview Score = Yüksek*

- Interview Score tarafında entropy veya gini index'e göre karar verildiğinde daha verimli bir DT olduğu sonucuna varılacaktır.
- Bunun sebebi GPA tarafında 2y/2n, 2y/1n gibi impure dağılımlar olması
- **Interview Score = Yüksek** ise → **Yes** tahmini,
- Aksi halde → **No** tahmini.

Tahminler

ID	GPA	Interview Score	Approval (Y)	Tahmin	Tahmin Doğru mu?
1	<3.0	Düşük	No	No	Yes
2	<3.0	Yüksek	Yes	Yes	Yes
3	<3.0	Yüksek	Yes	Yes	Yes
4	>3.0	Düşük	No	No	Yes
5	>3.0	Yüksek	Yes	Yes	Yes
6	>3.0	Normal	Yes	No	No
7	<3.0	Normal	No	No	Yes

Table 2: Sadece ID 6 yanlış tahmin edilecektir

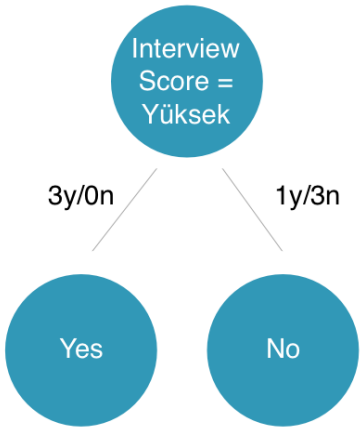

Figure 3: Interview Score Tree Stump Adayı

7 Adım 2: Hata Oranı ve Ağırlık Hesabı

Yanlış sınıflanan satırlar: ID 6.

Hesaplamalar

Başlangıçta tüm örnekler eşit ağırlıkla başlar: $w_i = \frac{1}{7} \approx 0.143$

Hata oranı:

$$\epsilon = \frac{\text{Yanlış sınıflanan toplam ağırlık}}{\text{Toplam ağırlık}} = \frac{0.143}{1} = 0.143$$

Ağırlık katsayısı:

$$\alpha = \frac{1}{2} \ln\left(\frac{1-\epsilon}{\epsilon}\right) = \frac{1}{2} \ln\left(\frac{1-0.143}{0.143}\right) \approx 0.895$$

- İlk hesaplamadan önce tüm row'lara aynı ağırlık verilir. Örneğin 7 row olduğu için 1/7.
- İlk hesaplama sonrasında sadece 1 hata yapıldığını görüyoruz. Buradan hata oranını belirtilen formüle ölçüyoruz ve ağırlık katsayısını 0.895 olarak buluyoruz. Bu ağırlık oranı artık bu decision tree'nin kullanacağı ağırlık katsayısı
- Bir sonraki adımda diğer DT'ye aktarılması için row'ların ağırlık katsayılarını da güncelleyeceğiz

8 Adım 3: Ağırlıkların Güncellenmesi

Güncelleme Formülleri

- Doğru sınıflananlar için:

$$w_i' = w_i \times e^{-\alpha} = 0.143 \times e^{-0.895} \approx 0.058$$

- Yanlış sınıflananlar için:

$$w_i' = w_i \times e^{\alpha} = 0.143 \times e^{0.895} \approx 0.354$$

Yanlış sınıflandırılan row'lar daha yüksek ağırlıklarla tekrar sınıflandırılıyor. Böylece bir sonraki DT'ye aktarılırken seçilme olasılığını çok daha yüksek yapıyoruz. Çünkü bir sonraki DT'ye aktarılırken bin aralığı denilen bir mantıkla hareket edecek ve rastgele 0-1 arasında ürettiği sayılardan seçmeye çalışacak. Fakat önce bu ağırlıkların toplamı 1 (yüzde yüz) yapmadığı için normalizasyon yapacağız.

ID	Doğru mu?	Yeni Ağırlık (w')
1	Yes	0.058
2	Yes	0.058
3	Yes	0.058
4	Yes	0.058
5	Yes	0.058
6	No	0.354
7	Yes	0.058

Table 3: Güncellenmiş ağırlıklar (Adım 3 Sonrası)

9 Adım 4: Normalizasyon

Toplam:

$$T = (6 \times 0.058) + 0.354 = 0.348 + 0.354 = 0.702$$

Normalleştirilmiş ağırlıklar:

$$\text{Doğru sınıflananlar: } \frac{0.058}{0.702} \approx 0.083$$

$$\text{Yanlış sınıflanan (ID 6): } \frac{0.354}{0.702} \approx 0.504$$

Normalize edilen ağırlıklara bakılarak bin aralıkları oluşturulur.

ID	Doğru mu?	Normalleştirilmiş Ağırlık (w")
1	Yes	0.083
2	Yes	0.083
3	Yes	0.083
4	Yes	0.083
5	Yes	0.083
6	No	0.504
7	Yes	0.083

Table 4: Normalleştirilmiş ağırlıklar (Adım 4 Sonrası)

10 Adım 5: Bin Ataması

Bin aralıkları şu şekilde hesaplanır (toplamsal şekilde):

ID	Normalleştirilmiş Ağırlık	Bin Aralığı
1	0.083	0.000 - 0.083
2	0.083	0.083 - 0.166
3	0.083	0.166 - 0.249
4	0.083	0.249 - 0.332
5	0.083	0.332 - 0.415
6	0.504	0.415 - 0.919
7	0.083	0.919 - 1.000

Table 5: Bin aralıkları (Adım 5 Sonrası)

11 Adım 6: İkinci Stump ve Tahminler

Şimdi güncellenmiş veri kümesi ile ikinci stump seçilir. Bu seçilme esnasında 0-1 arasında rastgele sayılar oluşturulup iterasyon yapılır. Örneğin 7 kere seçilen 0-1 arasında bir değerin 0.415 - 0.919 arasına düşme olasılığı diğerlerinden çok daha yüksektir. Muhtemelen 7 değer seçilirse 3-4 tanesi daha önce yanlış sınıflandırılan değerimiz olacaktır. Bin aralığında en büyük ağırlığa sahip olan ID 6 yüksek olasılıkla yeniden seçilir, bu nedenle ikinci stump ID 6 üzerine yoğunlaşabilir.

Örneğin ikinci stump şöyle olsun:
GPA >3.0 mı?

- Evet → Yes,
- Hayır → No.

Bu stump'ın yeni tahminleri ve doğruluk durumu ayrı tabloda gösterilebilir.

ID	GPA	Interview Score	Approval (Y)	Tahmin	Doğru mu?
1	<3.0	Düşük	No	No	Yes
2	<3.0	Yüksek	Yes	No	No
3	<3.0	Yüksek	Yes	No	No
4	>3.0	Düşük	No	Yes	No
5	>3.0	Normal	Yes	Yes	Yes
6	>3.0	Normal	Yes	Yes	Yes
7	>3.0	Normal	Yes	Yes	Yes

Table 6: İkinci stump sonuçları: GPA >3.0 mı?

Tabloda görüldüğü gibi yanlış tahmin edilen row 3 kez seçilmiş (rastgele oluşturulan 0-1 arasındaki rakamlar sonucunda gayet olası bir sonuç) ve tekrar başka bir DT'de training'e girecektir.

12 Adım 7: Hata Oranı ve Ağırlık Hesabı (İkinci Stump)

Yanlış sınıflanan satırlar: ID 2, 3, 4.

Hesaplamalar

Toplam ağırlık:

$$\epsilon = \sum(\text{yanlış sınıflananların ağırlıkları}) = 0.083 + 0.083 + 0.083 = 0.249$$

Ağırlık katsayısı:

$$\alpha_2 = \frac{1}{2}\ln\left(\frac{1-0.249}{0.249}\right) = \frac{1}{2}\ln(3.016) \approx 0.552$$

13 Adım 8: Nihai Tahmin Fonksiyonu

AdaBoost nihai tahmini:

$$F(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x)$$

- $\alpha_1 \approx 0.895$ (Interview Score stump),
- $\alpha_2 \approx 0.552$ (GPA stump),
- $h_m(x) = +1$ (Yes), -1 (No).

14 Örnek: Yeni Test Verisi

Test verisi:

- GPA = 3.2,
- Interview Score = Yüksek.

İlk stump tahmini:

Interview Score = Yüksek → Yes (+1).

İkinci stump tahmini:

GPA >3.0 → Yes (+1).

Hesaplama:

$$F(x) = (0.895)(+1) + (0.552)(+1) = 0.895 + 0.552 = 1.447$$

Sonuç:

$$\text{sign}(F(x)) = +1 \rightarrow Yes$$

15 Örnek: Diğer Test Verisi

Test verisi:

- GPA = 2.8,
- Interview Score = Yüksek.

İlk stump tahmini:

Interview Score = Yüksek → Yes (+1).

İkinci stump tahmini:

GPA >3.0 mı? Hayır → No (−1).

Hesaplama:

$$F(x) = (0.895)(+1) + (0.552)(-1) = 0.895 - 0.552 = 0.343$$

Sonuç:

$$\text{sign}(F(x)) = +1 \rightarrow Yes$$

Burada ilk DT'nin ağırlığı daha yüksek olduğu için ikinciden No çıksa da genel sonuç Yes'e dönüyor.

16 Şema: Nihai Model Akışı

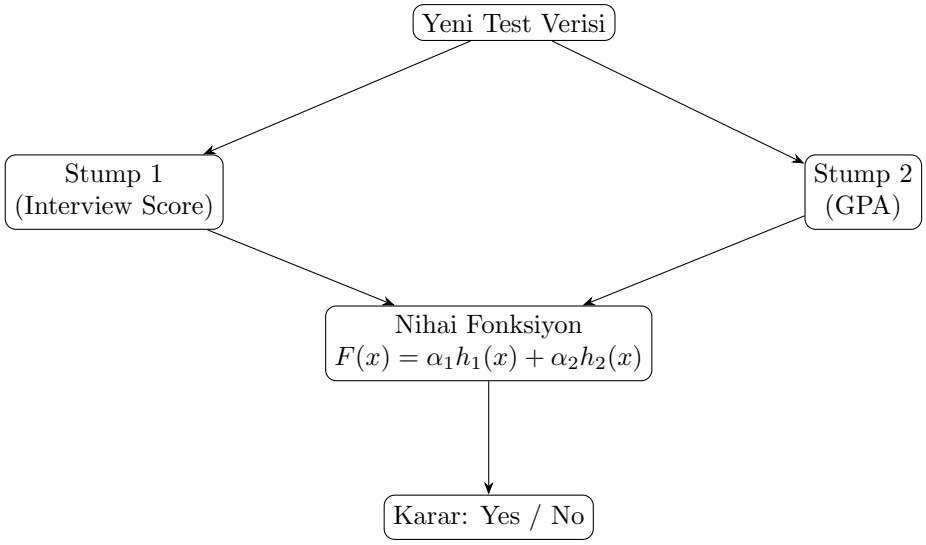

Figure 4: AdaBoost nihai karar akışı

17 AdaBoost Regressor: Regresyon için AdaBoost

AdaBoost yalnızca sınıflandırma problemleri için değil, aynı zamanda regresyon problemleri için de kullanılabilir. **AdaBoost Regressor**, temel olarak zayıf regresyon modellerini (örneğin derinliği düşük karar ağaçları) ardışık olarak eğitir ve bu modellerin hatalarına göre ağırlıklandırılmış bir tahmin oluşturur.

Sınıflandırma ve Regresyon Arasındaki Fark

- Sınıflandırma versiyonunda tahminler sınıf etiketleridir (örneğin Yes/No), regresyon versiyonunda ise sayısal değerlerdir.
- Sınıflandırmada log-loss ve exponential loss gibi fonksiyonlar kullanılırken, AdaBoost Regressor genellikle **square loss** veya **absolute loss** (L2 veya L1) kullanır.
- Her iterasyonda hedef Y yerine artık (residual) tahmin edilir.

Regresyon Temel Fikir

AdaBoost Regressor'ın amacı:

$$F(x) = \sum_{m=1}^{M} \alpha_m h_m(x)$$

fonksiyonu ile x için bir regresyon tahmini üretmektir.

Her iterasyonda:

1. $r_i^{(m)} = y_i - F_{m-1}(x_i)$ residual (artık) değeri hesaplanır.
2. $h_m(x)$ modeli, residual değerleri tahmin etmek için eğitilir.
3. $F_m(x) = F_{m-1}(x) + \alpha_m h_m(x)$ ile model güncellenir.

Not: Eğer $\alpha_m = 1$ alınırsa, bu durumda model direkt olarak her yeni öğreniciyi $F(x)$ fonksiyonuna ekler. Bazı implementasyonlarda bu katsayı hata oranlarına göre öğrenilir.

18 Örnek Veri Seti

ID	Experience (Years)	Salary (k$)
1	1	40
2	2	45
3	3	50
4	4	58
5	5	60
6	6	65

Table 7: AdaBoost Regressor için örnek veri seti

19 Adım 1: İlk Model ve Residual Hesabı

İlk model genellikle basit bir karar ağacı veya sabit bir tahmin olabilir. Örneğin:

$$F_0(x) = \text{Ortalama Salary} = \frac{40 + 45 + 50 + 58 + 60 + 65}{6} = 53$$

Gerçek Y	Tahmin $\mathbf{F_0(x)}$	Residual ($r_1 = Y - F_0(x)$)
40	53	-13
45	53	-8
50	53	-3
58	53	+5
60	53	+7
65	53	+12

Table 8: İlk model tahminleri ve residual hesapları

20 Adım 2: Residual'lara Uygun Yeni Model

Yeni model $h_1(x)$, residual değerlerini tahmin etmeye çalışır. Basit bir karar ağacı, residual'lara göre bölünerek aşağıdaki gibi olabilir:

$$h_1(x) = \begin{cases} -10 & \text{if Experience } < 3 \\ +8 & \text{otherwise} \end{cases}$$

Güncellenmiş model:

$$F_1(x) = F_0(x) + \alpha_1 h_1(x)$$

Burada α_1 genellikle 1 alınır, ya da optimize edilebilir.

21 Adım 3: Yeni Residual ve İterasyon Devamı

Yeni residual'lar:

$$r_2 = y_i - F_1(x_i)$$

Her iterasyonda bu adımlar devam eder. Amaç, artıkların sıfıra yaklaşmasıdır. Yani model artık hataları öğrenmeyi bırakana kadar eğitim devam eder.

22 Sonuç

AdaBoost Regressor:

- Zayıf regresyon modellerinin artık hatalarını öğrenerek güçlü hale gelir.
- Her adımda residual'ları hedef olarak alır.
- Sınıflandırmaya göre daha yumuşak ve sayısal çıktılar üretir.

Boosting algoritmalarının regresyon versiyonları genellikle *robust* ve *ensemble* yöntemler olarak kullanılır. AdaBoost Regressor, özellikle veri setindeki gürültülere duyarlıdır, bu nedenle genellikle **gradient boosting** gibi yöntemlere kıyasla daha az tercih edilse de eğitim açısından faydalı bir yapı sunar.

23 Sonuç

Bu çalışmada AdaBoost algoritmasının adım adım nasıl ilerlediğini; veri setinden başlayıp zayıf öğreniciler oluşturma, ağırlık hesaplama, güncelleme ve nihai tahmin üretme süreçlerini detaylandırdık. Basit bir örnek veri seti üzerinden süreci gözlemleyerek matematiksel hesaplamaları ve karar verme mekanizmalarını anlamış olduk. AdaBoost, doğru karar ağaçlarının seçilmesi ve hataların dengelenmesiyle güçlü bir sınıflandırıcı üretir.