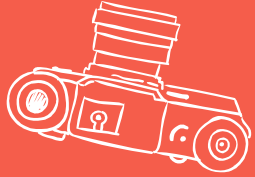
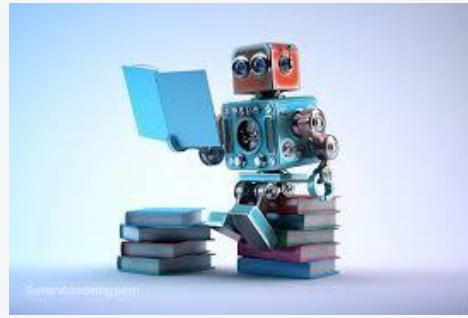


به نام خدا



یادگیری ماشین





یادگیری ماشین

آرش عبدی هجراندوست

arash.abdi.hejrandoost@gmail.com

دانشگاه علم و صنعت
دانشکده مهندسی کامپیوتر
نیم سال اول ۱۴۰۱-۱۴۰۲

حضرت درخت تصمیم (🌿)

❌ یکی از ساده ترین و موفق ترین گونه های یادگیری ماشین

❌ یک درخت تصمیم، بازنمایی از یک تابع است که

○ ورودی = برداری از ویژگی ها

○ خروجی = یک مقدار خروجی = تصمیم

❌ برای سادگی در اینجا فرض می کنیم ورودی گسسته و خروجی دودویی است.

○ ورودی و خروجی می توانند پیوسته باشند.

❌ خروجی دودویی:

○ نمونه های Positive

○ نمونه های Negative

تصمیم گیری

❌ درخت تصمیم برای رسیدن به تصمیم (خروجی) تعدادی از ویژگی‌های ورودی را در نظر می‌گیرد

❌ هر گره در درخت تصمیم متناظر با یکی از ویژگی‌های ورودی (A_i) است.

❌ شاخه‌های انشعاب یافته از هر گره برپسبسی متناظر با مقادیر ممکن برای آن ویژگی دارند

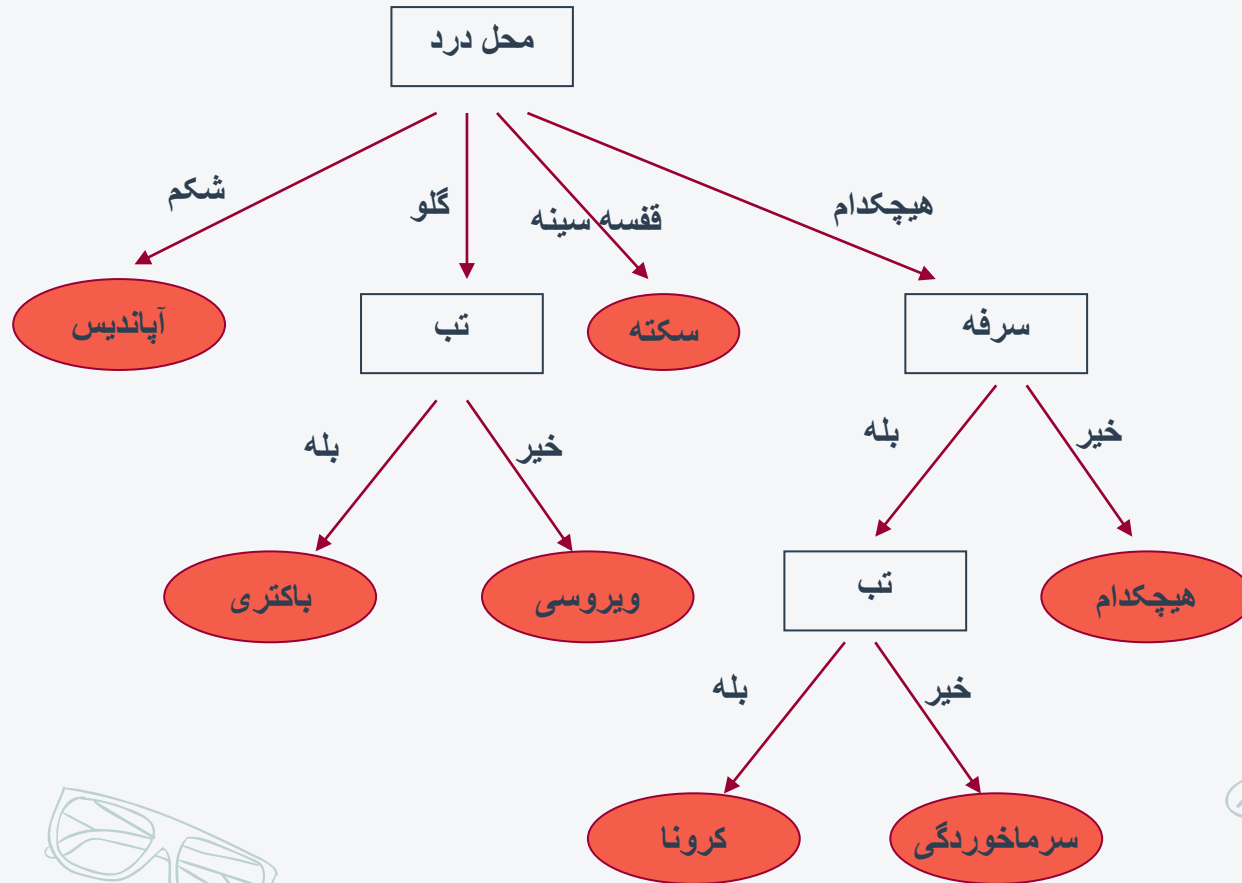
$$A_i = v_{ik}$$

❌ هر برگ نیز یک مقدار دارد که مشخص کننده خروجی درخت است

❌ این نحوه بازنمایی تناسب با نحوه بیان انسان دارد.

❌ بسیاری از دفترچه‌های راهنما (مثلا برای تعمیر خودرو) طبق ساختار یک درخت تصمیم نوشته می‌شوند

مثالی از درخت تصمیم



مثالی دیگر

❌ درختی برای اینکه تصمیم بگیریم آیا در یک رستوران منتظر خالی شدن میز بمانیم یا خیر؟

○ هدف = فروجی = پیش بینی مقدار فروجی تابع WillWait بر اساس ویژگی‌های ورودی
○ ورودی ها:

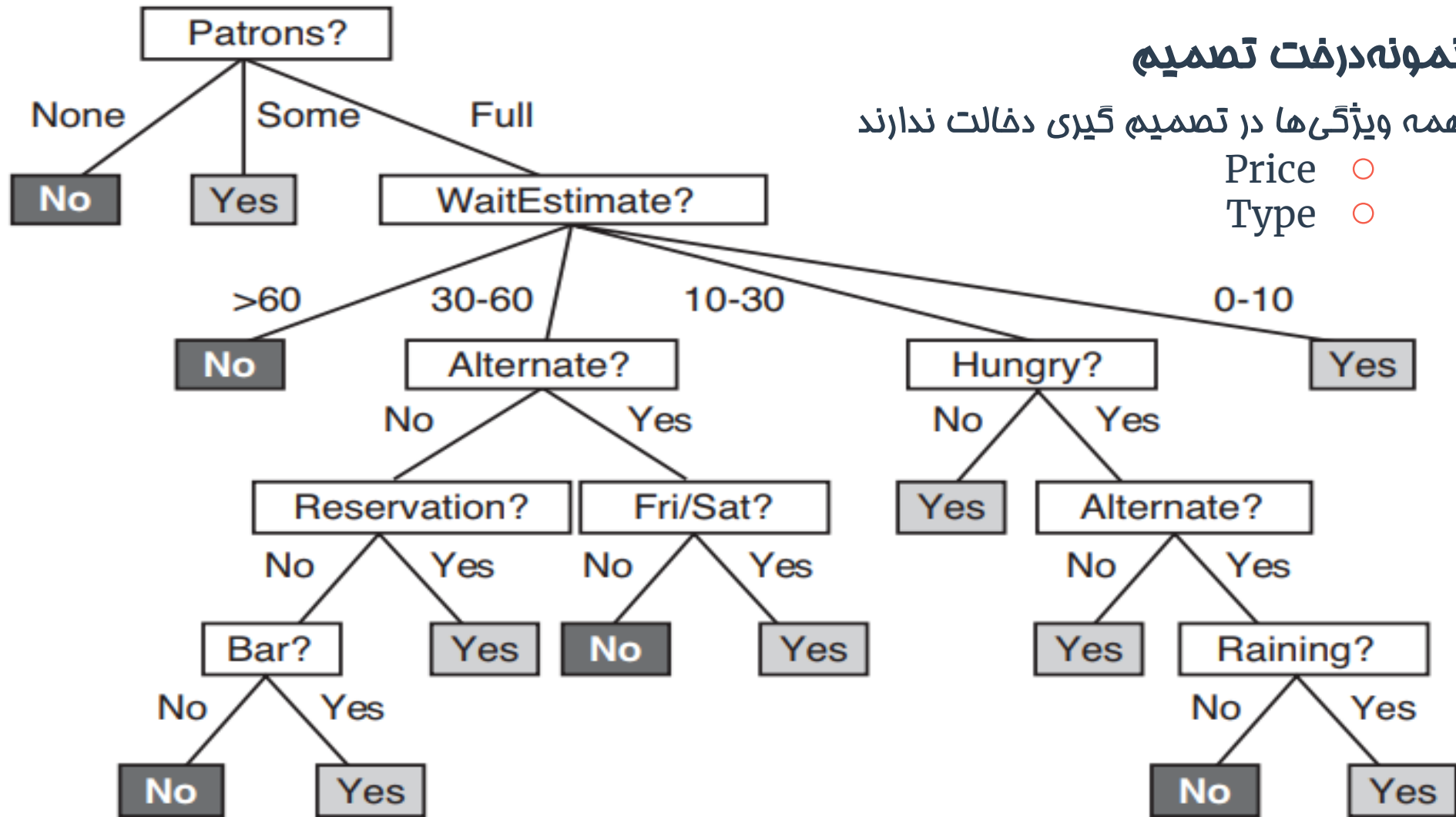
- Alternate
- Bar (Just for Wait!)
- Fri/Sat
- Hungry
- Patrons (None, Some, Full)
- Price
- Raining
- Reservation
- Type (French, Italian, Thai, burger)
- WaitEstimate (0-10 min, 10-30, 30-60, >60)

نمونه درخت تصمیم

همه ویژگی‌ها در تصمیم گیری دخال دارند

Price ○

Type ○



معنای درخت تصمیم

✗ یک درخت تصمیم دودوئی، منطقاً معادل آن است که گفته شود:

○ ویژگی خروجی True است اگر و فقط اگر ویژگی‌های ورودی، یکی از مسیرهای ریشه تا برگی با مقدار True را ارضا کند.

○ $Goal \Leftrightarrow (Path1 \vee Path2 \vee \dots)$

○ هر مسیر، ترکیب عطفی تست روی مقدار-ویژگی‌هایی است که در طول مسیر وجود دارد
■ تست: اگر ویژگی مربوطه مقدار زیرشافه این مسیر را دارد، خروجی = True و در غیر این صورت
False = خروجی

✗ این نحوه بازنمایی را Disjunctive Normal Form (DNF) می‌نامند.

○ ترکیب فصلی عبارات عطفی (برعکس CNF)

○ آیا هر جمله در منطق گزاره‌ای را می‌توان به صورت DNF نوشت؟ چرا؟

○ اگر چنین باشد، هر جمله در منطق گزاره‌ای را می‌توان با یک درخت تصمیم نشان داد.

✗ در مورد بسیاری از مسائل، ساختار درخت تصمیم بیانی موجز و مناسب از مساله را ارائه می‌کند

✗ اما در برخی توابع، ممکن درخت بزرگی به وجود بیاید.
○ مثلاً تابع Majority

■ فروجی یک اگر بیشتر از نصف ورودی‌ها یک باشند و فروجی صفر در غیر این صورت

■ لازم است اغلب یا تمام ورودی مشاهده شود

■ حجم درخت به طور نمایی بالا میرود (برماسب عمق)

✗ بنابراین درخت تصمیم در برخی مسائل کارا و در برخی ناکارآمد است

○ مانند هر بازنمایی دیگری

✗ فرض کنید تعداد n ویژگی دودویی داشته باشیم. تعداد توابع دودویی چقدر است؟

- تعداد حالات ورودی = تعداد سطرهای جدول درستی = 2^n
- ستون خروجی این جدول تابعی با 2^n ورودی را تشکیل می‌دهد ←
تعداد توابع قابل تعریف 2^{2^n}
- تعداد درختهای ممکن بیشتر از 2^{2^n} است زیرا درختهایی که معنای یکسان ولی شکل متفاوت داشته باشند نیز وجود دارد
- جایجا کردن ترتیب مشاهده ویژگی‌ها
- مثلاً برای ۱۰ ویژگی مثال رستوران، تعداد توابع ممکن منجر به عددی ۱۰۹ رقمی می‌شود
- بنابراین باید به دنبال روشی برای یافتن درخت/فرضیه مناسب باشیم

ساخت درخت تصمیم از روی نمونه

× نمونه‌هایی با فرمت (X, y)

○ X = برداری از مقادیر ویژگی‌های ورودی

○ Y = یک مقدار دودویی خروجی

× مجموعه‌ای ۱۲ تایی از این نمونه‌ها به عنوان مجموعه آموزشی در صفحه بعد نشان داده شده است.

× نمونه‌ها به دو دسته مثبت و منفی (بر اساس خروجی) تقسیم می‌شوند

× به دنبال درختی باید بود که:

○ سازگار با نمونه‌های موجود باشد

○ تا حد ممکن کوچک باشد

Example	Input Attributes										Goal
	<i>Alt</i>	<i>Bar</i>	<i>Fri</i>	<i>Hun</i>	<i>Pat</i>	<i>Price</i>	<i>Rain</i>	<i>Res</i>	<i>Type</i>	<i>Est</i>	<i>WillWait</i>
\mathbf{x}_1	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>French</i>	<i>0–10</i>	$y_1 = \text{Yes}$
\mathbf{x}_2	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>30–60</i>	$y_2 = \text{No}$
\mathbf{x}_3	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>Some</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Burger</i>	<i>0–10</i>	$y_3 = \text{Yes}$
\mathbf{x}_4	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Thai</i>	<i>10–30</i>	$y_4 = \text{Yes}$
\mathbf{x}_5	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Full</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>French</i>	<i>>60</i>	$y_5 = \text{No}$
\mathbf{x}_6	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$</i>	<i>Yes</i>	<i>Yes</i>	<i>Italian</i>	<i>0–10</i>	$y_6 = \text{Yes}$
\mathbf{x}_7	<i>No</i>	<i>Yes</i>	<i>No</i>	<i>No</i>	<i>None</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Burger</i>	<i>0–10</i>	$y_7 = \text{No}$
\mathbf{x}_8	<i>No</i>	<i>No</i>	<i>No</i>	<i>Yes</i>	<i>Some</i>	<i>\$\$</i>	<i>Yes</i>	<i>Yes</i>	<i>Thai</i>	<i>0–10</i>	$y_8 = \text{Yes}$
\mathbf{x}_9	<i>No</i>	<i>Yes</i>	<i>Yes</i>	<i>No</i>	<i>Full</i>	<i>\$</i>	<i>Yes</i>	<i>No</i>	<i>Burger</i>	<i>>60</i>	$y_9 = \text{No}$
\mathbf{x}_{10}	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$\$\$</i>	<i>No</i>	<i>Yes</i>	<i>Italian</i>	<i>10–30</i>	$y_{10} = \text{No}$
\mathbf{x}_{11}	<i>No</i>	<i>No</i>	<i>No</i>	<i>No</i>	<i>None</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Thai</i>	<i>0–10</i>	$y_{11} = \text{No}$
\mathbf{x}_{12}	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Yes</i>	<i>Full</i>	<i>\$</i>	<i>No</i>	<i>No</i>	<i>Burger</i>	<i>30–60</i>	$y_{12} = \text{Yes}$

یادگیری درخت تصمیم

✗ یافتن کوچکترین درخت سازگار با جستجوی تمام درخت‌های ممکن، غیرممکن/غیرعملیاتی است.

✗ با ایده‌های خلاقانه می‌توان درختی به اندازه کافی خوب، یافت.
○ کوچک، اما نه لزوماً کوچکترین درخت

✗ الگوریتم یادگیری درخت تصمیم، روشی مریضانه برای این منظور ارائه می‌کند.
○ همیشه مهمترین ویژگی را زودتر تست کن!
○ مساله را به ساختن چند زیر درخت کوچکتر تقسیم کن.
○ ادامه بده.

✗ مهمترین ویژگی؟
○ ویژگی‌ای که بیشترین تمایز را بین نمونه‌ها برای دسته‌بندی ایجاد می‌کند

1	3	4	6	8	12
2	5	7	9	10	11

Type?

French

1
5

Italian

6
10

Thai

4	8
2	11

Burger

3	12
7	9

(a)

1	3	4	6	8	12
2	5	7	9	10	11

Patrons?

None

7	11

No

Some

1	3	6	8

Yes

Full

4	12		
2	5	9	10

Hungry?

No

5	9

Yes

4	12
2	10

(b)

Type ویژگی فوبی نیست / Patrons نسبتاً ویژگی مهمی است

بعد از انتخاب هر ویژگی، هر زیرشافه آن ویژگی، درختی کوچکتر تشکیل می‌دهد

یادگیری درخت تصمیم در زیر درخت‌ها مستقلاً باید ادامه بیابد

نحوه تصمیم گیری در زیرشاخه‌ها

- ✗ اگر تمام نمونه‌های زیرشاخه مثبت (منفی) باشند، جواب معلوم است (Yes یا No)
- ✗ اگر تعدادی نمونه مثبت و تعدادی منفی مانده است، بهترین ویژگی بعدی باید انتخاب شود و نمونه‌ها مجدداً به چند دسته تقسیم شوند.
- ✗ اگر در زیرشاخه‌ای، هیچ نمونه داده‌ای وجود نداشت:
 - نمونه داده‌ای برای این ترکیب ویژگی‌ها وجود ندارد
 - می‌توان بر اساس وضعیت تمامی نمونه‌ها در گره والد تصمیم گرفت
 - اکثریت نمونه‌ها در گره والد
- ✗ اگر ویژگی جدیدی وجود نداشت ولی هم نمونه مثبت و هم نمونه منفی موجود بود:
 - نمونه‌هایی با ویژگی یکسان ولی خروجی متفاوت داریم.
 - چرا؟ (ویژگی بیشتر؟ خطا؟)
 - بر اساس اکثریت نمونه‌های موجود در زیرشاخه تصمیم می‌گیریم.

function DECISION-TREE-LEARNING(*examples*, *attributes*, *parent_examples*) **returns**
a tree

if *examples* is empty **then return** PLURALITY-VALUE(*parent_examples*)
else if all *examples* have the same classification **then return** the classification
else if *attributes* is empty **then return** PLURALITY-VALUE(*examples*)
else

$A \leftarrow \operatorname{argmax}_{a \in \text{attributes}} \text{IMPORTANCE}(a, \text{examples})$

tree \leftarrow a new decision tree with root test *A*

for each value v_k of *A* **do**

$\text{exs} \leftarrow \{e : e \in \text{examples} \text{ and } e.A = v_k\}$

subtree \leftarrow DECISION-TREE-LEARNING(*exs*, *attributes* – *A*, *examples*)

add a branch to *tree* with label (*A* = v_k) and subtree *subtree*

return *tree*

❌ خروجی الگوریتم یادگیری درخت تصمیم بستگی زیادی به نمونه‌های ورودی دارد

❌ در شکل زیر خروجی الگوریتم برای مثال رستوران با ۱۲ نمونه داده آمده است.

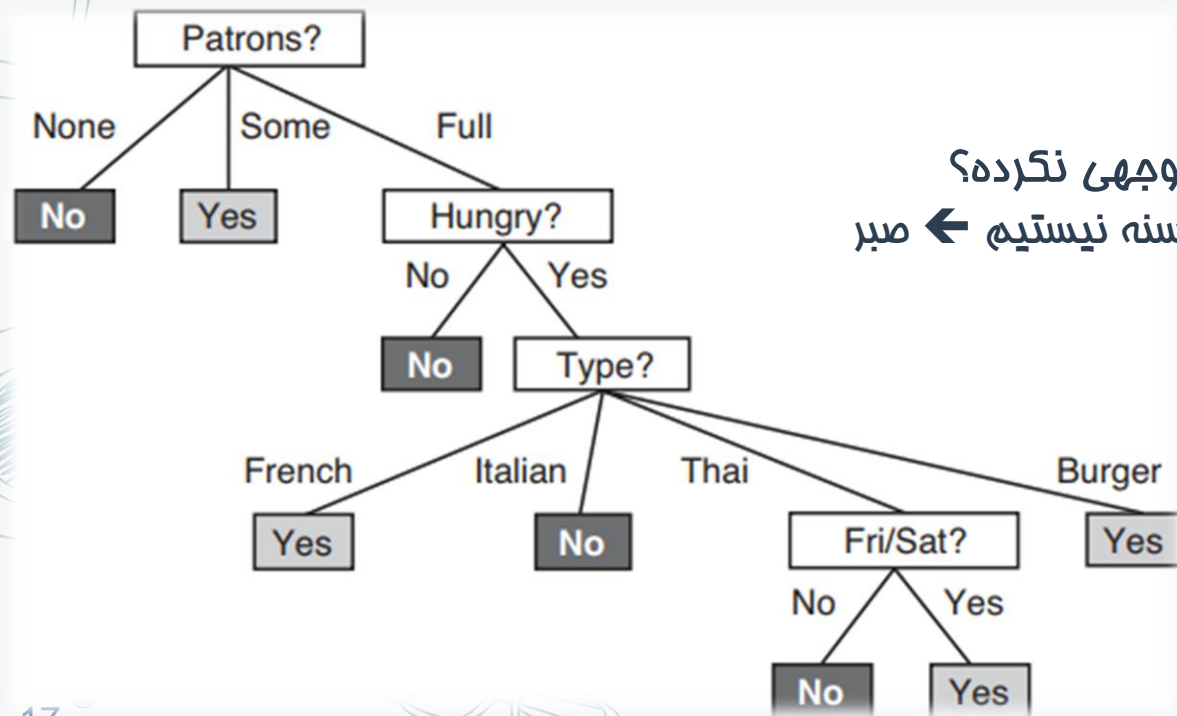
❌ نتیجه خوبی است؟

○ البته کوچک است!

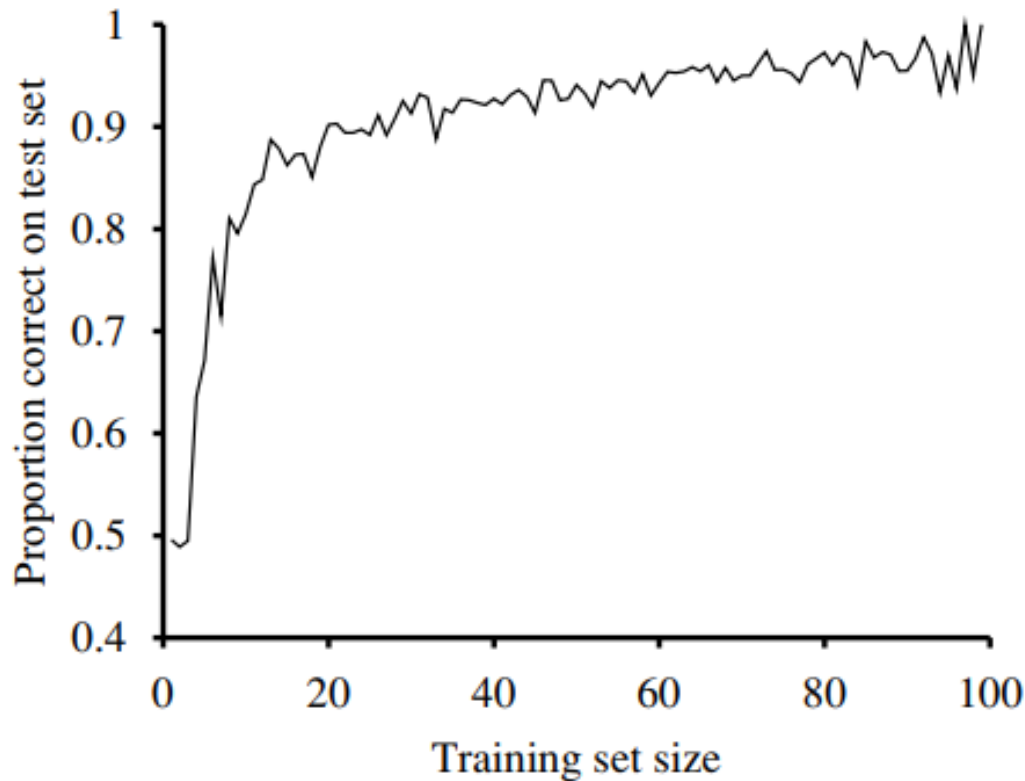
❌ چرا به زمان انتظار زیر ۱۰ دقیقه توجهی نکرده؟
○ مثلاً رستوران Full است و گرسنه نیستیم ← صبر نکن! حتی ۱۰ دقیقه!

❌ راه حل؟

○ مجموعه نمونه‌ای دیگر
○ تعداد نمونه‌ها



ارزیابی درخت تصمیم



مجموعه آزمایشی 

منحنی یادگیری 

۱۰۰ نمونه داده تصادفی تولید

می‌کنیم

تقسیم به مجموعه آموزشی و


آزمایشی

تعداد نمونه‌های آموزشی از ۱ تا ۹۹

متغیر


۲۰ بار تکرار هر آزمایش برای

متوسط‌گیری

وجود عنصر تصادف در الگوریتم؟ 

وقتی که چند ویژگی با اهمیت

برابر داریم.

تعداد نمونه آموزشی چند باشد؟ 

انتخاب ویژگی در درخت تصمیم

✗ انتخاب ویژگی با هدف کوچک شدن درخت نهایی

- ایده: انتخاب مهمترین ویژگی در هر لحظه (مریضانه)
- ویژگی که بیشترین تمایز را ایجاد می‌کند

✗ باید معیاری عددی برای اهمیت (خوب بودن) ویژگی ها بیان کرد

✗ معیار:

- بی نظمی – آنتروپی (Entropy)
- دست‌آورد اطلاعات (Information Gain)

عالی جناب، آنتروپی

- ✗ آنتروپی معیاری است برای سنجش میزان عدم قطعیت یک متغیر تصادفی
- ✗ متغیر تصادفی که تنها یک مقدار ممکن دارد (سکه ای که همیشه خط می‌آید)، هیچ عدم قطعیتی ندارد
 - آنتروپی = صفر
 - مشاهده مقدار آن هیچ «اطلاعاتی» به دست نمی‌دهد.
- ✗ متغیری که دو مقدار با احتمال برابر دارد (سکه سالم)، «یک بیت (bit)» آنتروپی دارد
- ✗ طاسی با چهار وجه دارای «دو بیت» آنتروپی است
 - دو بیت لازم است تا نشان دهیم کدام وجه طاس مشاهده شده است.
- ✗ سکه ای که ۹۹٪ موارد خط می‌آید:
 - آنتروپی نزدیک به صفر

تعریف آنتروپی

برای یک متغیر تصادفی V با مقادیر ممکن V_k و احتمال هر مقدار برابر با $P(V_k)$ مقدار عددی آنتروپی به این شکل محاسبه میشود:

$$\text{Entropy: } H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)} = - \sum_k P(v_k) \log_2 P(v_k)$$

$$H(\text{Fair}) = -(0.5 \log_2 0.5 + 0.5 \log_2 0.5) = 1$$

آنتروپی سکه سالم:

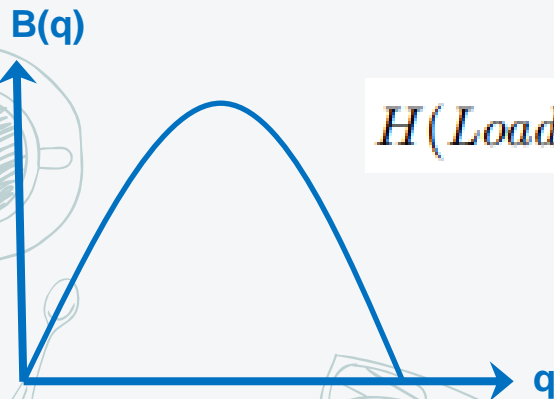
آنتروپی سکه ای با احتمال ۹۹٪ خطا:

$$H(\text{Loaded}) = -(0.99 \log_2 0.99 + 0.01 \log_2 0.01) \approx 0.08 \text{ bits.}$$

آنتروپی متغیر دودویی با احتمال q برای True بودن:

$$B(q) = -(q \log_2 q + (1 - q) \log_2 (1 - q))$$

$$H(\text{Loaded}) = B(0.99) \approx 0.08.$$



Entropy: $H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)} = - \sum_k P(v_k) \log_2 P(v_k)$

متغیری با d مقدار ممکن ✖

کمترین مقدار آنتروپی: احتمال یک مقدار برابر با ۱ و احتمال بقیه مقادیر برابر صفر $\leftarrow \text{Min}(H) = 0$ ○

بیشترین مقدار آنتروپی: احتمال مساوی برای هر مقدار $(p=1/d)$ $\leftarrow \text{Max}(H) = \log_2(d)$ ○

اگر در ساختن درخت تصمیم، مجموعه داده آموزشی دارای p نمونه مثبت و n نمونه منفی باشد، داریم: ✖

$$H(\text{Goal}) = B\left(\frac{p}{p+n}\right)$$

Goal متغیر تصادفی است که تعلق نمونه‌ها به کلاس مثبت یا منفی را مشخص می‌کند. ✖

در مثال رستوران با ۱۲ نمونه آموزشی، $p=n=6$ \leftarrow آنتروپی: $B(0.5)=1 \text{ bit}$ ✖

داده‌ها دو حالت دارند: مثبت یا منفی \leftarrow کلا ۱ بیت اطلاعات قابل دست یافتن است، اگر $p=n$ باشد (وگرنه؟) ○

هر ویژگی که تست (مشاهده) شود، مقداری از این ۱ بیت اطلاعات را می‌تواند ارائه کند ○ دانشگاه، علم، صنعت

کدام ویژگی؟

یک ویژگی مشخص A با d مقدار ممکن، مجموعه داده‌های آموزشی E را به d زیر مجموعه E_1, E_2, \dots, E_d تقسیم می‌کند. ❌

هر زیر مجموعه E_k دارای p_k نمونه مثبت و n_k نمونه منفی است. ❌

اگر با مشاهده ویژگی A به زیر شاخه k ام برویم، به اندازه $B(p_k/(p_k+n_k))$ بیت اطلاعات اضافی لازم داریم تا بتوانیم جواب (وضعیت مثبت / منفی بودن نمونه) را پیدا کنیم. ❌

○ اگر p_k یا p_n صفر باشند، مقدار اطلاعات تکمیلی برای جواب لازم است؟

امتثال آنکه یک نمونه آموزشی دارای ویژگی A با مقدار k باشد، $(p_k+n_k)/(p+n)$ است. ❌

آنتروپی باقیمانده با تست/مشاهده ویژگی A برابر است با: ❌

$$Remainder(A) = \sum_{k=1}^d \frac{p_k+n_k}{p+n} B\left(\frac{p_k}{p_k+n_k}\right)$$

❌ بخش اول، وزن می‌دهد به آنتروپی هر زیرشاخه

$$Remainder(A) = \sum_{k=1}^d \frac{p_k + n_k}{p + n} B\left(\frac{p_k}{p_k + n_k}\right)$$

دست آورد اطلاعات INFORMATION GAIN

✗ دست آورد اطلاعات با مشاهده ویژگی A برابر است با میزان کاهش آنتروپی با مشاهده A :

$$Gain(A) = B\left(\frac{P}{p+n}\right) - Remainder(A)$$

✗ دست آورد اطلاعات می گوید با تست یک ویژگی مشخص چه مقدار به جواب نزدیک تر می شویم.

✗ دست آورد اطلاعات نمی تواند منفی شود (در بدترین حالت، صفر است. در چه شرایطی صفر است؟)

✗ برای محاسبه تابع اهمیت (Importance) در ساخت درخت تصمیم تنها به $Gain(A)$ نیاز داریم.

✗ مثلاً در مثال (ستوران):

$$Gain(Patrons) = 1 - \left[\frac{2}{12} B\left(\frac{0}{2}\right) + \frac{4}{12} B\left(\frac{4}{4}\right) + \frac{6}{12} B\left(\frac{2}{6}\right) \right] \approx 0.541 \text{ bits,}$$

$$Gain(Type) = 1 - \left[\frac{2}{12} B\left(\frac{1}{2}\right) + \frac{2}{12} B\left(\frac{1}{2}\right) + \frac{4}{12} B\left(\frac{2}{4}\right) + \frac{4}{12} B\left(\frac{2}{4}\right) \right] = 0 \text{ bits}$$

مقابله با بیش برازش - OVERFITTING

✗ هرس درخت:

✗ هدف: مقابله با بیش برازش

○ حذف گره‌های بی‌ربط

○ شروع از گره‌هایی که فقط فرزند برگ دارند.

○ اگر ویژگی تست شده در گره، بی‌ربط شناخته شود، این تست

هرس شده و به یادش یک برگ گذاشته می‌شود! (فرزندان گره

حذف شده و خود گره تبدیل به برگ می‌شود)

○ تکرار چرخه تا زمانی که چیزی برای هرس نماند.

معیار هرس

✗ تشخیص بی‌ربط بودن ویژگی تست شونده در یک گره؟

- گره‌ای با n نمونه منفی و p نمونه مثبت
- اگر ویژگی انتخاب شده، زیرشافه‌هایی ایجاد کند که نسبت نمونه مثبت و منفی در آنها تقریباً مانند همین گره باشد (یعنی $p/(p+n)$)
- از آنجا که طبق الگوریتم درخت تصمیم، این ویژگی مهمترین ویژگی بوده، انتخاب ویژگی دیگر کم فایده‌تر است و این گره باید برگ شود

✗ ویژگی بی‌ربط، دست‌آورد اطلاعات نزدیک به صفر دارد

○ معیار بی‌ربط بودن ویژگی: دست‌آورد اطلاعات کم. (آیا بالا بودن Remainder میانگین وزن دار آنتروپی) کافی است؟

$$Remainder(A) = \sum_{k=1}^d \frac{p_k + n_k}{p + n} B\left(\frac{p_k}{p_k + n_k}\right)$$

$$Gain(A) = B\left(\frac{p}{p+n}\right) - Remainder(A)$$

✗ هرس درخت باعث می‌شود نویز در داده‌ها تاثیر منفی کمتری داشته باشد.
○ ویژگی دارای نویز، دست‌آورد اطلاعات کمی دارد (نمونه‌ها را تصادفی به زیرشافه‌ها تقسیم می‌کند)

✗ درخت هرس شده کوچک تر و قابل فهم تر است.

هرس یا خاتمه زودهنگام؟

✗ چرا به جای هرس، در مین ساخت درخت، وقتی ویژگی دارای دست آورد اطلاعات بالا پیدا نکردیم، ساخت زیرشافه‌ها را متوقف نکنیم؟ (هزینه ساخت کمتر)

✗ ممکن است یک ویژگی خوب پیدا نشود، اما ترکیب چند ویژگی با هم دارای دست آورد اطلاعات باشند.

○ مثال: تابع XOR (ورودی دو بعدی و خروجی XOR)

○ ۱۰۰ نمونه داریم که به چهار زیرمجموعه ۲۵ تایی به شکل زیر تقسیم شده اند (آنتروپی = ۱)

■ دسته اول: ۰۰ (خروجی منفی)

■ دسته دوم: ۰۱ (خروجی مثبت)

■ دسته سوم: ۱۰ (خروجی مثبت)

■ دسته چهارم: ۱۱ (خروجی منفی)

○ انتخاب هر ویژگی در ریشه، دست آورد اطلاعات صفر خواهد داشت

○ اما در عمق دوم درخت، انتخاب ویژگی بعدی، دست آورد اطلاعات برابر ۱ دارد.

درخت تصمیم در داده‌های پیوسته

✗ در بسیاری از مواقع، داده‌های مسائل واقعی، پیوسته هستند.

✗ ورودی/ویژگی پیوسته

- گسسته سازی (وزن < 40 ، دمای بالای ۸۰ و ...)
- یافتن نقاط برش برای هر ویژگی ورودی
- مقادیر ورودی مربوطه برای داده‌های آموزشی را مرتب کن
- در جاهایی که برچسب فروجی تخییر میکند، برش بزن.

✗ هزینه گسسته سازی احتمالا گران ترین بخش ساخت درخت باشد

✗ خروجی پیوسته

- درخت رگرسیون (نه دسته بندی)
- در برگ‌ها، تابعی خطی برحسب ورودی‌ها (نه لزوما همه ورودی‌ها)، روی داده‌های موجود در برگ fit میشود و تعیین کننده خروجی نهایی در آن برگ خواهد بود.

■ تعبیر هندسی

- چگونه ویژگی مهم را انتخاب کنیم؟ (محاسبه آنتروپی بدون برچسب خروجی)
- ایده ۱: گسسته سازی خروجی با طول گام ثابت (مالا عملا برچسب خروجی داریم)
- ایده ۲: صرف نظر کردن از کوچک بودن درخت و انتخاب همه ورودی‌ها به ترتیب یافتن نقاط برش هر ویژگی بر اساس داده‌های آموزشی که در آن گره قرار گرفته اند (نه همه داده‌ها)

ایده ۳: استفاده از ایده ۲، ولی لزوماً تا آخرین ویژگی جلو نرویم. وقتی داده‌های موجود در یک زیر شافه خروجی‌های نزدیک به هم داشتند، دیگر ویژگی بعدی تست نشود و تابع خطی fit شود

ایده ۴: جایگزینی آنتروی با مفهومی دیگر که برای خروجی پیوسته همان مفهوم بی‌نظمی را داشته باشد

- مثلاً واریانس خروجی (ویژگی انتخاب شود که کمترین واریانس را در هر زیر شافه دارد)
- انتخاب ویژگی براساس معیارهای قبلی (دست‌آورد اطلاعات) با محاسبه واریانس به جای آنتروپی

ایده ۵؟



پروژه درخت تصمیم

۲ هفته



درباره تمرین‌ها، تمویل‌ها، گزارش‌نویسی و مقاله‌خوانی



خسته نباشیم!

با تشکر

