دانشگاه علم و صنعت
دانشکده مهندسی کامپیوتر

یادگیری از روی نمونه داده

«هوش مصنوعی: رهیافتی نوین»، فصل ۱۸

مدرس: آرش عبدی هجراندوست

نیم‌سال دوم ۱۴۰۱-۱۴۰۲

فهرست مطالب

- مقدماتی از یادگیری
- انواع یادگیری
- یادگیری با نظارت
- درخت تصمیم

یادگیری

- یکی از مهترین ویژگی‌های یک ماشین هوشمند
- تقلید از موجودات هوشمند طبیعت
  - شکل گیری دسته بندی در کودکی با دیدن نمونه
- کاربردی شدن بیشتر هوش مصنوعی
  - مجموعه ای از داده ها
  - الگوریتم یادگیری
  - پیش بینی خروجی

عامل یادگیرنده

- یک عامل یادگیرنده است اگر کارایی خود را به مرور زمان و با مشاهدات جدید بهبود دهد.
  - از افزایش کارایی در سرعت نوشتن یک شماره تلفن (یا زدن امضا)
  - تا ارائه یک نظریه جدید درباره جهان توسط انیشتین
- یک نوع پرطرفدار از یادگیری:
  - یادگیری تابعی که خروجی را تخمین میزند، با داشتن مجموعه ای از جفتهای ورودی-خروجی

لزوم یادگیری

- سوال: چرا عامل یاد بگیرد؟
  - چرا خود طراح عامل هر آنچه لازم است را ابتدائا به خورد عامل ندهد؟

- طراح ممکن است همه حالتها را نداند
- طراح ممکن است همه تغییرات محیطی در طول زمان را نداند
- طراح خیلی وقتها نمی‌داند که یک چیز را چگونه می‌داند، بنابراین نمی‌تواند برنامه‌اش را بنویسد:
  - شناخت اعضای خانواده از روی تصویر چهره

انواع یادگیری

- کدام مولفه از عامل باید بهبود پیدا کند؟
  - ادراک؟ استنتاج؟ ...
    - خودروی هوشمند: کی ترمز کند؟ آنچه می‌بیند کی اتوبوس است؟ تاثیر یک عمل در جاده خیس چیست؟ وقتی در آخر روز پول چندانی از مسافران گیرش نیامد، بفهمد که کلا کارا نبوده...
- عامل چه دانش اولیه‌ای دارد؟
- دانش چگونه بازنمایی شده است؟
- چه بازخوردی از محیط موجود است برای بهبود عامل؟

یادگیری استقرایی (Inductive)

- ورودی ها برداری از مقادیر ویژگی (Attribute) است
- خروجی ها : یک مقدار پیوسته یا گسسته
- یادگیری از از روی جفت نمونه های ورودی – خروجی یادگیری استقرایی نام دارد
  - یادگیری استنتاجی (Deductive) در مقابل یادگیری استقرایی است که در آن با کمک دانش و قواعد موجود دانش جدید کشف میشود.

بازخورد در یادگیری

- یادگیری غیرنظارتی (Unsupervised)
  - یادگیری الگو هایی از ورودی بدون دانش بازخورد (خروجی)
  - خوشه بندی
    - مثلا خودروی هوشمند ممکن است کم کم مفهومی به نام «روز خوب ترافیکی» یا «روز بد ترافیکی» را یادبگیرد بدون آنکه از کسی بازخوردی گرفته باشد (صرفا با حسگرها و ادراکات ورودی خودش)
- یادگیری تقویتی (Reinforcement)
  - بازخورد = پاداش یا جریمه برای هرعمل یا برای مجموعه ای از اعمال
    - تصادف در اعمال خودروی هوشمند
- **یادگیری با نظارت (Supervised)**
  - داشتن نمونه‌هایی از ورودی-خروجی و یادگرفتن تابعی که ورودی را به خروجی نگاشت کند.
    - مجموعه ای از تصاویر با تگ اتوبوس/نااتوبوس
    - مجموعه ای از ادراکات خودرو با تگ «ترمز کن» یا «بپیچ به چپ» که توسط انسان داده شده.

بازخورد در یادگیری

- یادگیری نیمه نظارتی (Semi-Supervised)
  - داشتن تعدادی داده با برچسب خروجی و تعدادی بدون خروجی
  - حتی خروجی ها ممکن است دقیق یا درست نباشند (نویز)
    - تخمین سن بر اساس چهره
    - دروغ گفتن سن توسط اشخاص یا نا دقیق گفتن آن
  - در هر دو صورت نویزی بودن خروجی یا فقدان خروجی، چیزی بین یادگیری با و بی نظارت داریم

یادگیری با نظارت

- مجموعه ای آموزشی از N نمونه داده با فرمت ورودی-خروجی به شکل زیر داریم:

$$(x_1, y_1), (x_2, y_2), \ldots (x_N, y_N)$$

که در آن مقادیر yi بر اساس تابعی مجهول به شکل $y=f(x)$ تولید شده اند.

وظیفه یادگیری با نظارت یافتن تابع h است که مقدار تابع صحیح (یعنی f) را تقریب می‌زند.

- در اینجا x و y میتوانند هر مقداری (حتی غیر عددی) داشته باشند.
- تابع h یک فرضیه (Hypothesis) نام دارد.
- یادگیری وظیفه دارد در بین فرضیه های ممکن، بهترین آنها را بیابد
  - بهترین در داده های آموزشی
  - و داده های آزمایشی

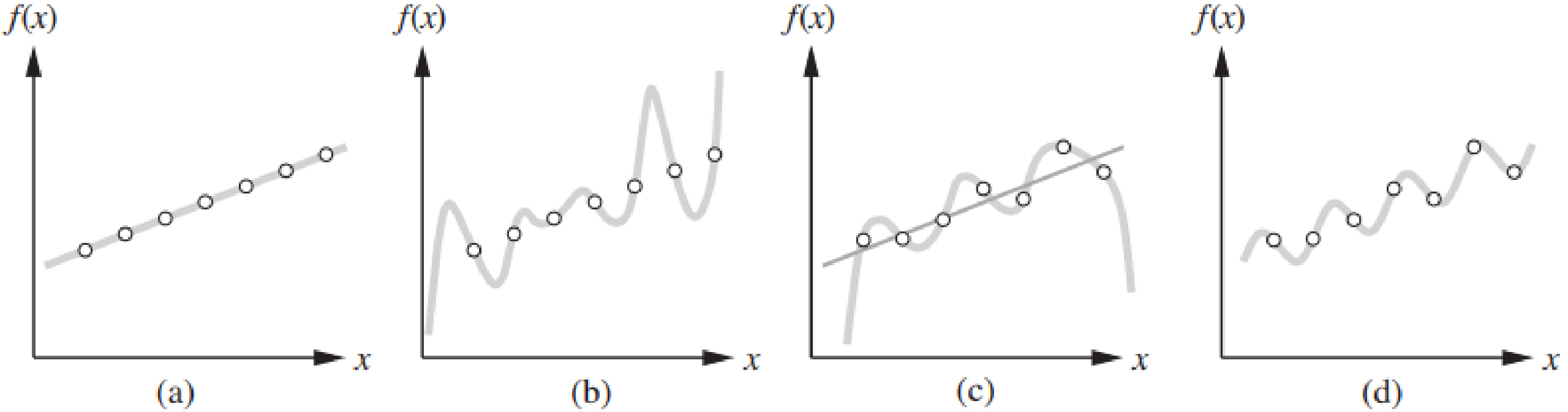
تعمیم (Generalization)

- برای سنجش دقت یک یادگیری، مجموعه داده آزمایشی (جدای از داده های آموزشی) مبنا قرار داده می‌شود.
- اگر فرضیه پیدا شده بتواند در داده های آزمایشی هم کارا باشد، آن فرضیه دارای تعمیم است.
  - کارایی: پیش بینی مقدار Y از روی X

دسته‌بندی و تقریب

- اگر خروجی y مقداری از یک مجموعه محدود داشته باشد (مثلا بارانی-ابری-آفتابی)، مساله یادگیری، یک دسته بندی نام خواهد داشت
  - و اگر تنها دو مقدار خروجی ممکن باشد، دسته بندی دودوئی (Boolean-Binary)
- اگر خروجی y یک عدد (نامحدود) باشد (مثلا دمای فردا)، مساله یادگیری، Regression نام خواهد داشت.
  - به طور دقیق، رگرسیون شامل یافتن امیدریاضی یا متوسط مقدار y (برای هر ورودی) می‌باشد، زیرا احتمال یافتن مقدار دقیق خروجی صفر است.
    - آنچه مشاهده شده، صرفا تعدادی نمونه است و با استقرا به دنبال فهمیدن حقیقت هستیم.
    - **حتی اگر** مقدار خروجی برای یک ورودی معین x در همه داده های مشاهده شده، دقیقا یک عدد y باشد، امیدریاضی خروجی برای x، y است، اما احتمال آنکه خروجی برای ورودی x بعدی (مشاهده نشده) دقیقا y باشد، صفر است! (مفهوم حد)
    - احتمال آنکه خروجی مقداری در یک بازه داشته باشد، میتواند غیر صفر باشد، اما احتمال عدد دقیق برای خروجی صفر (حدی) است.

نمونه های از رگرسیون



- انتخاب تابع h از فضای فرضیه های H
- فرضیه سازگار: با نمونه های موجود منطبق باشد

انتخاب فرضیه

- انتخاب از بین چند فرضیه سازگار؟ (در H)
- برقراری توازن بین:
  - سادگی ← تعمیم
  - پیچیدگی ← دقت
- توسعه فضای فرضیه H به سینوسی در شکل قبل (بخش d)
  - تامین سادگی (تعداد پارامتر کم) و دقت، با هم
- اهمیت فضای فرضیه
- یک مساله یادگیری امکان پذیر یا شدنی است اگر فضای فرضیه شامل تابع درست باشد.

انتخاب فرضیه در یادگیری با نظارت

- با داشتن نمونه های داده (data)، انتخاب فرضیه در یادگیری بانظارت به شکل زیر قابل بیان است:

$$h^* = \underset{h \in \mathcal{H}}{\operatorname{argmax}} P(h|data)$$

- یا

$$h^* = \underset{h \in \mathcal{H}}{\operatorname{argmax}} P(data|h)\, P(h)$$

- با این فرمول، میتوان گفت احتمال P(h) برای تابع چند جمله ای درجه ۱ یا ۲، بالا و برای چندجمله ای درجه های بالاتر، کم است.
- با دادن احتمال پایین برای چند جمله ای های درجه بالا، اجازه می‌دهیم خروجی یادگیری تابعی پیچیده باشد، وقتی که داده‌ها واقعا نیاز به چنان تابع پیچیده‌ای داشته باشند.
- در واقع بخش P(h) برای توابع پیچیده نقش جریمه را بازی می‌کند.

فضای فرضیه

- با توجه به اهمیت فضای فرضیه، چرا فضای فرضیه را به تمامی فرضیه های ممکن توسعه ندهیم؟
  - تمامی فرضیه های ممکن: تمام برنامه های کامپیوتری قابل نوشتن یا یک ماشین تورینگ
  - مطمئن باشیم فرضیه درست در مجموعه H وجود دارد
- پیچیدگی یافتن فرضیه درست
- برقراری توازن بین:
  - میزان رسا بودن و منطبق بودن بودن فضای فرضیه با داده ها (Expressiveness)
  - پیچیدگی/سادگی یافتن یک فرضیه در آن فضا
- فرضیه ساده تر:
  - یافتن ساده تر
  - استفاده کم هزینه تر

- برقراری توازن بین پیچیدگی/سادگی فرضیه و میزان رسا بودن فضای فرضیه، چندان ساده نیست:
  - گاهی فضای فرضیه رسا می‌تواند سبب یافتن یک فرضیه ساده (و در عین حال، سازگار) نیز بشود.
  - گاهی محدود کردن میزان رسا بودن فضا، سبب می‌شود هر فرضیه سازگاری لزوما پیچیده باشد

- مثلا، حذف sin از فضای فرضیه (محدود کردن میزان رسا بودن)، سبب میشود فرضیه چندجمله ای به شکلی بسیار پیچیده (درجه خیلی بالا) پیدا شود (با فرض آنگه سازگاری آن ممکن باشد)

- مثلا، منطق مرتبه اول رساتر/پیچیده تر/قدرتمندتر از منطق گزاره‌ای است.
  - قواعد بازی شطرنج با کمک منطق مرتبه اول می‌تواند در یکی دو صفحه نوشته شود
  - قواعد بازی شطرنج با کمک منطق گزاره ای (که ساده‌تر از منطق مرتبه اول است) چندصد/هزار صفحه حجم خواهد داشت.

حضرت درخت تصمیم ()

- یکی از ساده ترین و موفق ترین گونه‌های یادگیری ماشین
- یک درخت تصمیم، بازنمایی از یک تابع است که
  - ورودی = برداری از ویژگی‌ها
  - خروجی = یک مقدار خروجی = تصمیم
- برای سادگی در اینجا فرض می‌کنیم ورودی گسسته و خروجی دودوئی است.
  - ورودی و خروجی می‌توانند پیوسته باشند.
- خروجی دودوئی:
  - نمونه‌های Positive
  - نمونه‌های Negetive

تصمیم گیری

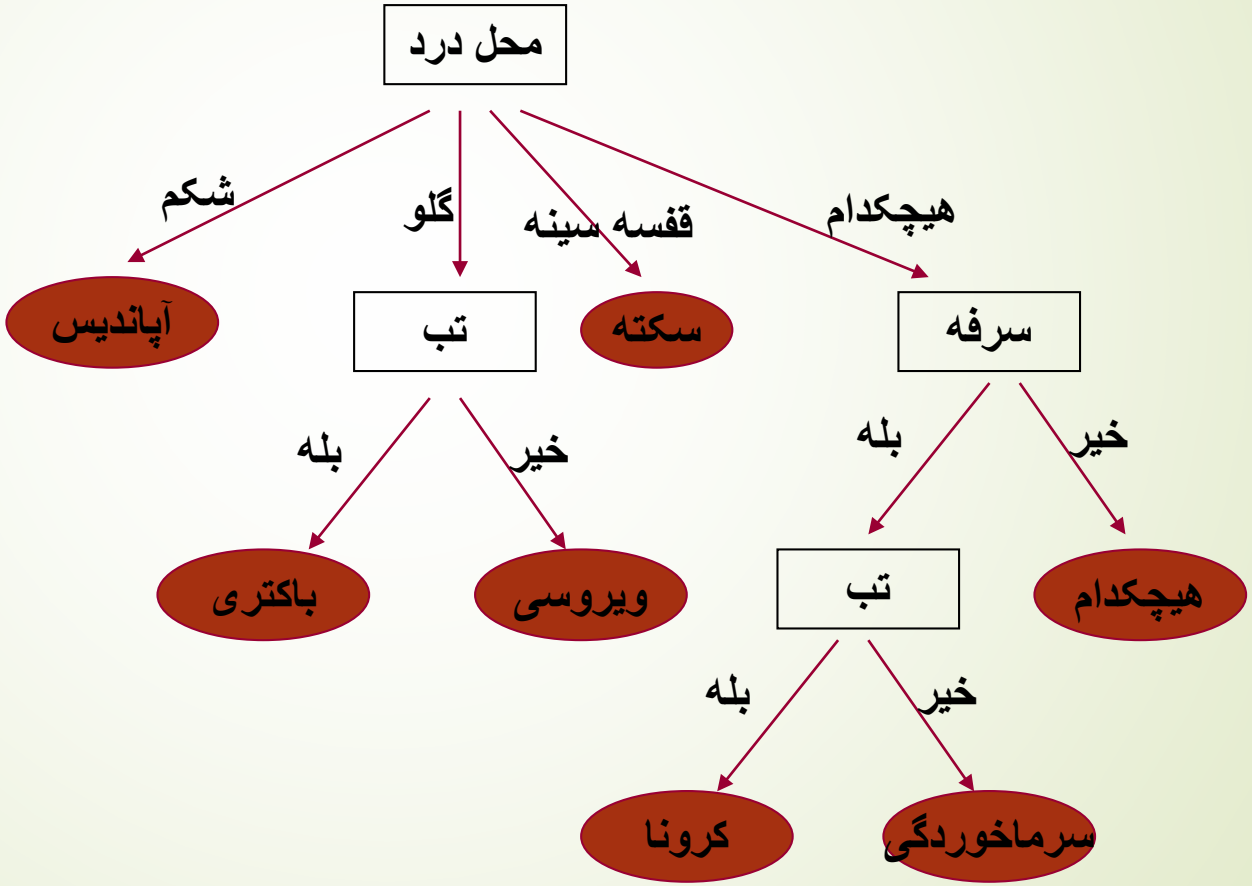
- درخت تصمیم برای رسیدن به تصمیم (خروجی) تعدادی از ویژگی‌های ورودی را در نظر می‌گیرد
- هر گره در درخت تصمیم متناظر با یکی از ویژگی‌های ورودی (A_i) است.
- شاخه‌های انشعاب یافته از هر گره برچسبی متناظر با مقادیر ممکن برای آن ویژگی دارند

$$A_i = v_{ik}$$

- 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

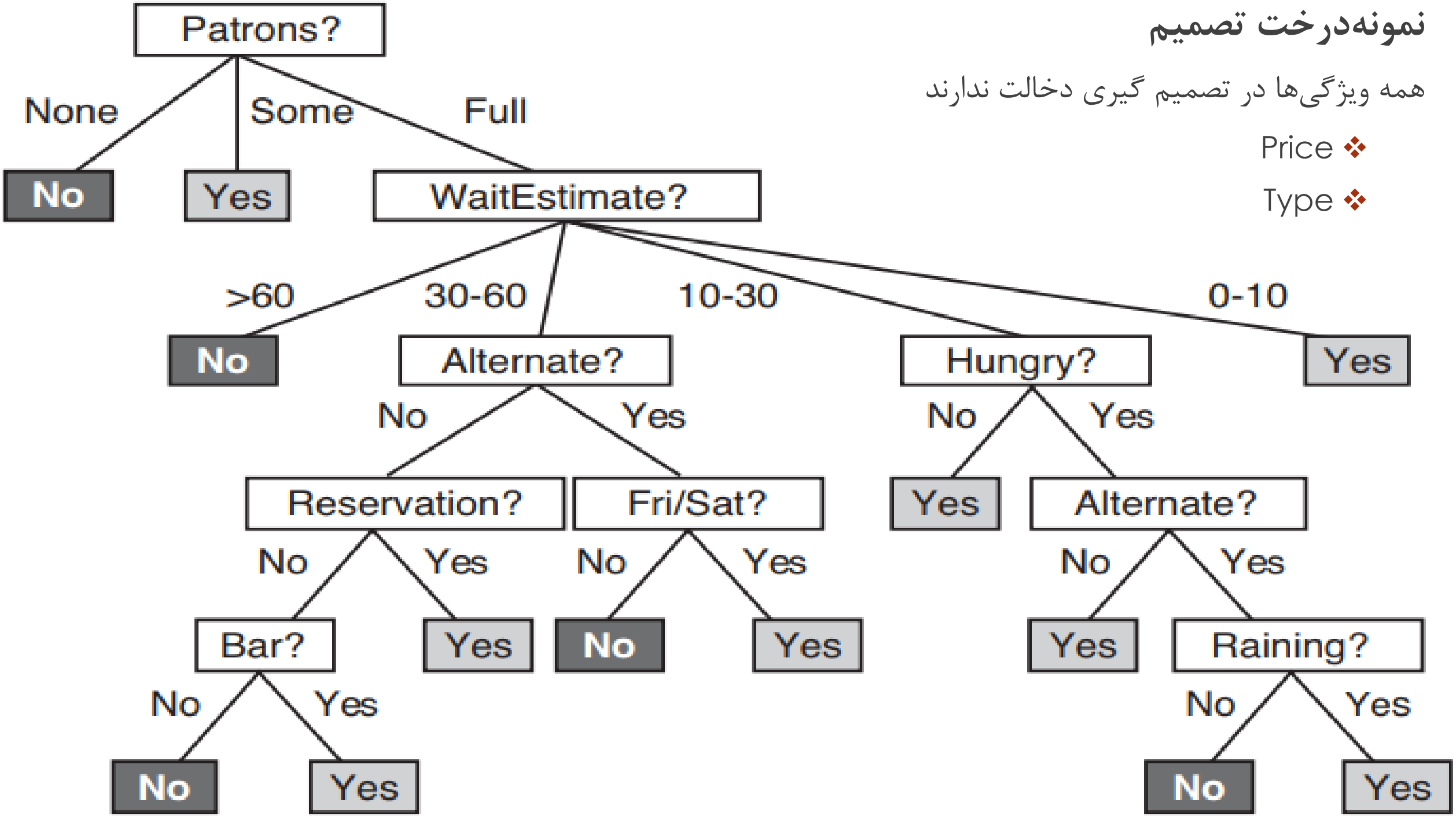
- درختی برای اینکه تصمیم بگیریم آیا در یک رستوران منتظر خالی شدن میز بمانیم یا خیر؟
  - هدف = خروجی = پیش بینی مقدار خروجی تابع WillWait بر اساس ویژگی‌های ورودی
  - ورودی ها:
    - Alternate
    - Bar (Just for Wait!)
    - Fri/Sat
    - Hungry
    - Patrons (None,Some,Full)
    - Price
    - Raining
    - Reservation
    - Type (French,Italian,Thai,burger)
    - WaitEstimate (0-10 min, 10-30, 30-60, >60)

دقت شود که ورودی ها دارای حالات محدودی هستند.

نمونه‌درخت تصمیم

همه ویژگی‌ها در تصمیم گیری دخالت ندارند

- Price
- Type



معنای درخت تصمیم

- یک درخت تصمیم دودوئی، منطقا 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.

$$Goal \Leftrightarrow (Path1 \vee Path2 \vee \cdots)$$

  - هر مسیر، ترکیب عطفی تست روی مقدار-ویژگی‌هایی است که در طول مسیر وجود دارد
    - تست: اگر ویژگی مربوطه مقدار زیرشاخه این مسیر را دارد، خروجی = True و در غیر این صورت خروجی = False
- 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 گزاره ای 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.
  - مثلا تابع Majority
    - خروجی یک اگر بیشتر از نصف ورودی‌ها یک باشند و خروجی صفر در غیر این صورت
    - لازم است اغلب یا تمام ورودی مشاهده شود
    - حجم درخت به طور نمایی بالا میرود (برحسب عمق)
- بنابراین درخت تصمیم در برخی مسائل کارا و در برخی ناکارآمد است
  - مانند هر بازنمایی دیگری
- فرض کنید تعداد n ویژگی دودوئی داشته باشیم. تعداد توابع دودوئی 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
    - جابجا کردن ترتیب مشاهده ویژگی‌ها
  - مثلا برای ۱۰ویژگی مثال رستوران، تعداد توابع ممکن منجر به عددی ۱۰۹ رقمی می‌شود
  - بنابراین باید به دنبال روشی برای یافتن درخت/فرضیه مناسب باشیم

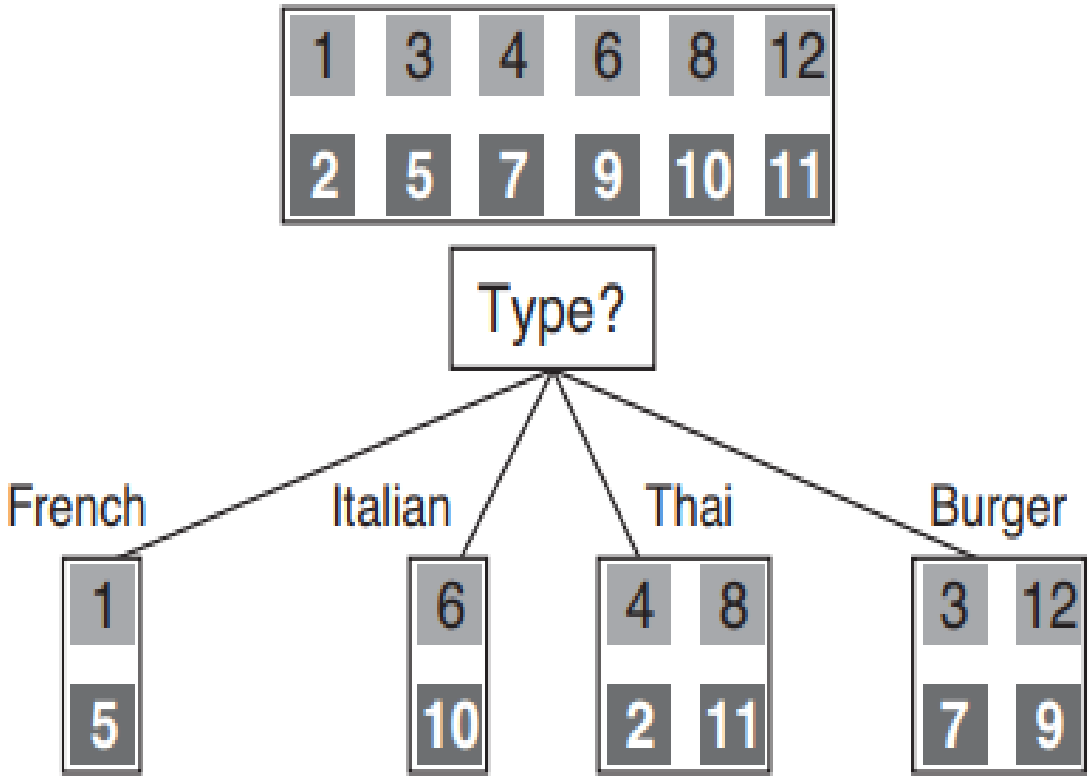
ساخت درخت تصمیم از روی نمونه

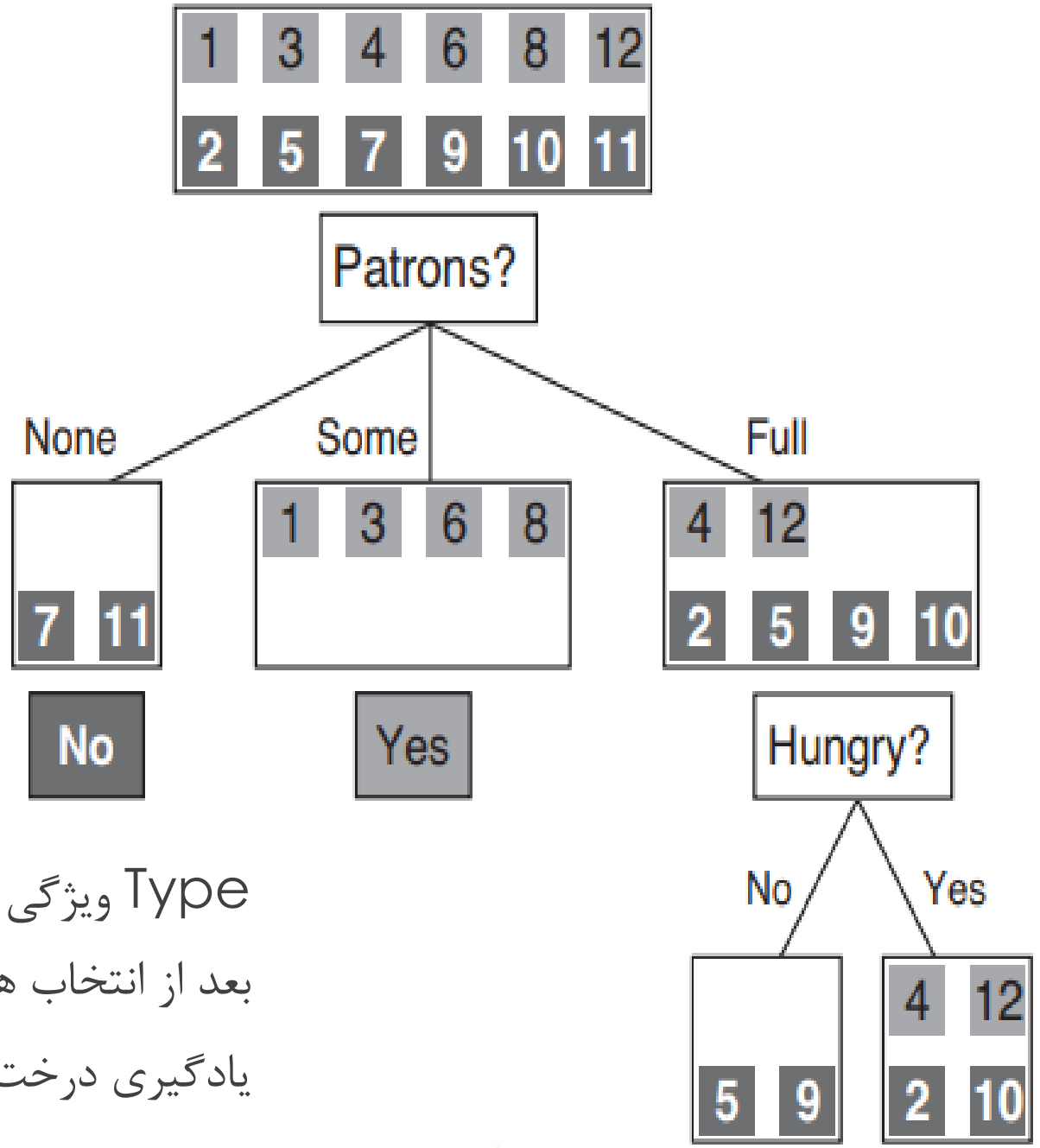
- نمونه‌هایی با فرمت (X,Y)
  - X = برداری از مقادیر ویژگی های ورودی
  - Y = یک مقدار دودوئی خروجی
- مجموعه ای ۱۲ تایی از این نمونه ها به عنوان مجموعه آموزشی در صفحه بعد نشان داده شده است.
- نمونه‌ها به دو دسته مثبت و منفی (بر اساس خروجی) تقسیم می‌شوند
- به دنبال درختی باید بود که:
  - سازگار با نمونه‌های موجود باشد
  - تا حد ممکن کوچک باشد

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

یادگیری درخت تصمیم

- یافتن کوچکترین درخت سازگار با جستجوی تمام درخت‌های ممکن، غیرممکن/غیرعملیاتی است.
- با ایده های خلاقانه می‌توان درختی به اندازه کافی خوب، یافت.
  - کوچک، اما نه لزوما کوچکترین درخت
- الگوریتم یادگیری درخت تصمیم، روشی حریصانه برای این منظور ارائه می‌کند.
  - همیشه مهمترین ویژگی را زودتر تست کن!
  - مساله را به ساختن چند زیر درخت کوچکتر تقسیم کن.
  - ادامه بده.
- مهترین ویژگی؟
  - ویژگی‌ای که بیشترین تمایز را بین نمونه ها برای دسته بندی ایجاد می‌کند





Type ویژگی خوبی نیست / Patrons نسبتا ویژگی مهمی است

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

یادگیری درخت تصمیم در زیر درخت‌ها مستقلا باید ادامه بیابد

(a) (b)

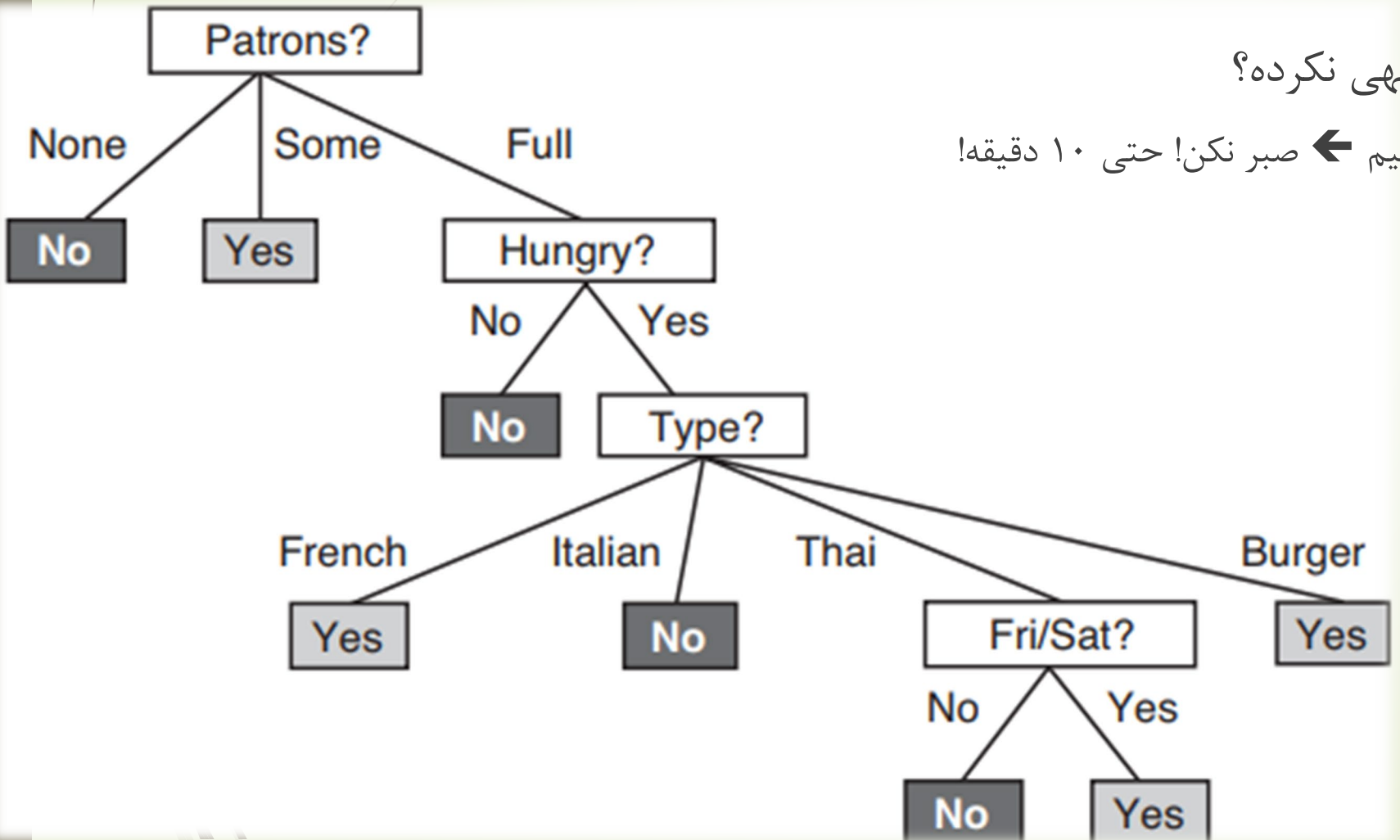
نحوه تصمیم گیری در زیرشاخه‌ها

- اگر تمام نمونه‌های زیرشاخه مثبت (یا منفی) باشند، جواب معلوم است (Yes یا No)
- اگر تعدادی نمونه مثبت و تعدادی منفی مانده است، بهترین ویژگی بعدی باید انتخاب شود و نمونه ها مجددا 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.

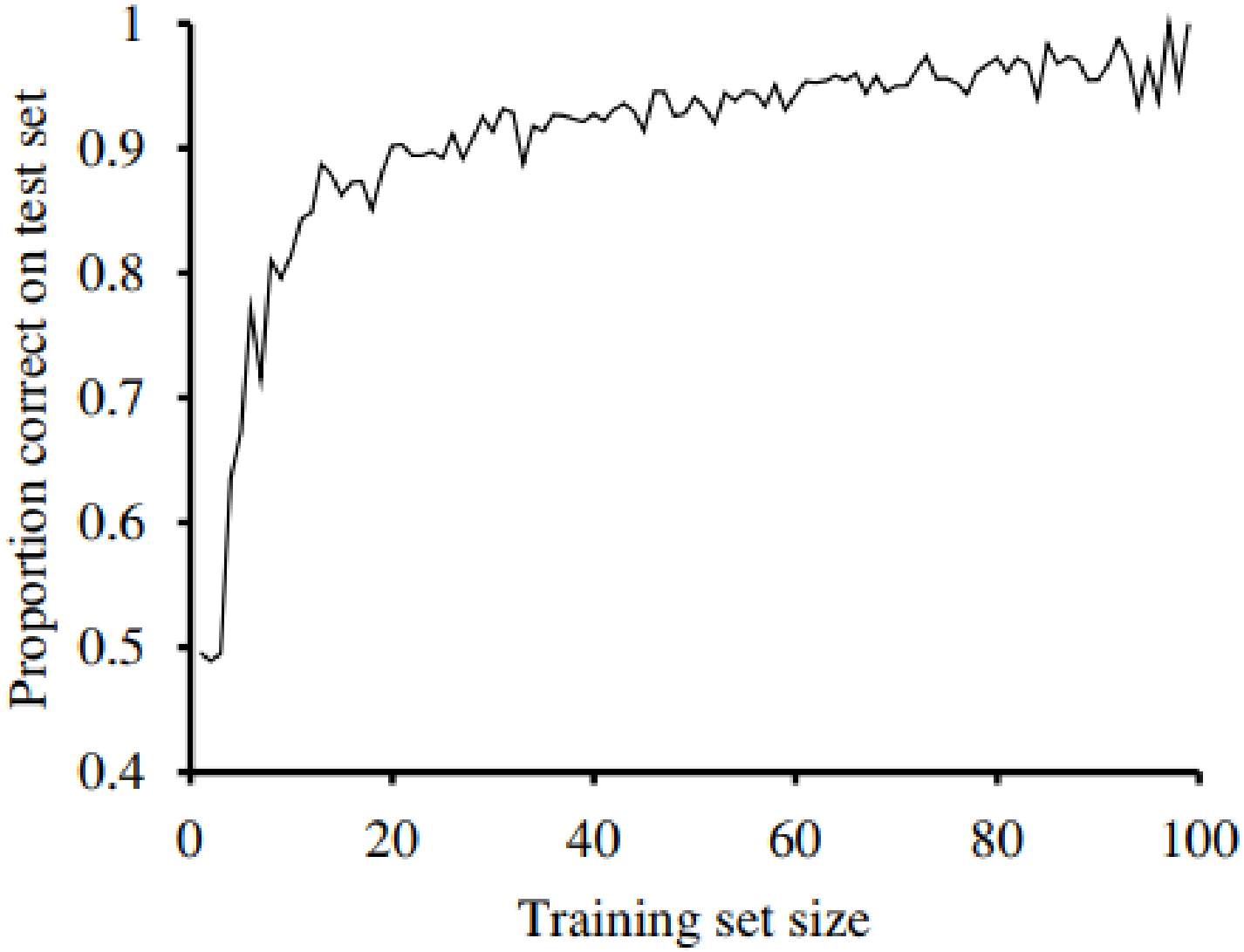
function DECISION-TREE-LEARNING(*examples*, *attributes*, *parent_examples*) **returns** a tree

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
  $A \leftarrow \text{argmax}_{a \in attributes}$ IMPORTANCE(a, *examples*)
  tree $\leftarrow$ a new decision tree with root test A
  for each value v_k of A **do**
    $exs \leftarrow \{e : e \in examples \text{ and } e.A = v_k\}$
    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؟
  - مثلا 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 (حریصانه)
  - 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)

عالی‌جناب، 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:

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

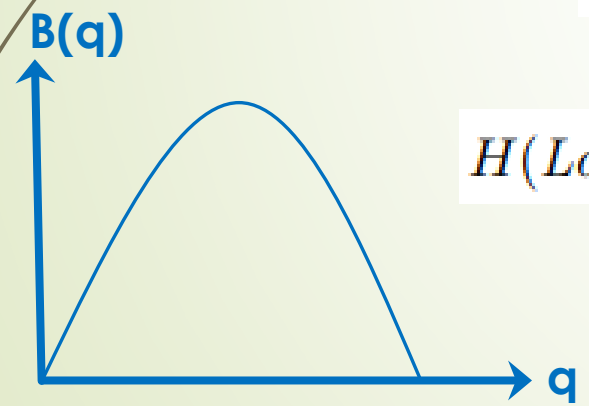
- آنتروپی سکه سالم:

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

- آنتروپی سکه ای با احتمال ۹۹٪ خط:

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

- آنتروپی متغیر دودوئی با احتمال q برای True بودن:



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

$$H(Loaded) = B(0.99) \approx 0.08$$

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 ← $Min(H) = 0$
  - بیشترین مقدار آنتروپی: احتمال مساوی برای هر مقدار (p=1/d) ← $Max(H) = log_2(d)$

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

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

- Goal متغیر تصادفی است که تعلق نمونه‌ها به کلاس مثبت یا منفی را مشخص می‌کند.
- درمثال رستوران با ۱۲ نمونه آموزشی، p=n=6 ← آنتروپی: B(0.5)=1 bit
  - داده ها دو حالت دارند: مثبت یا منفی ← 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

کدام ویژگی؟

- یک ویژگی مشخص A با d مقدار ممکن، مجموعه داده‌های آموزشی E را به d زیر مجموعه $E_1, E_2, \ldots, E_d$ 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، چقدر اطلاعات تکمیلی برای جواب لازم است؟
- **احتمال** 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

دست‌آورد اطلاعات
Information Gain

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

- دست‌آورد اطلاعات با مشاهده ویژگی A برابر است با میزان کاهش آنتروپی با مشاهده A:

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

- دست‌آورد اطلاعات می‌گوید با تست یک ویژگی مشخص چه مقدار به جواب نزدیک**تر** می‌شویم.
- دست‌آورد اطلاعات نمی‌تواند 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.
- مثلا در مثال رستوران:

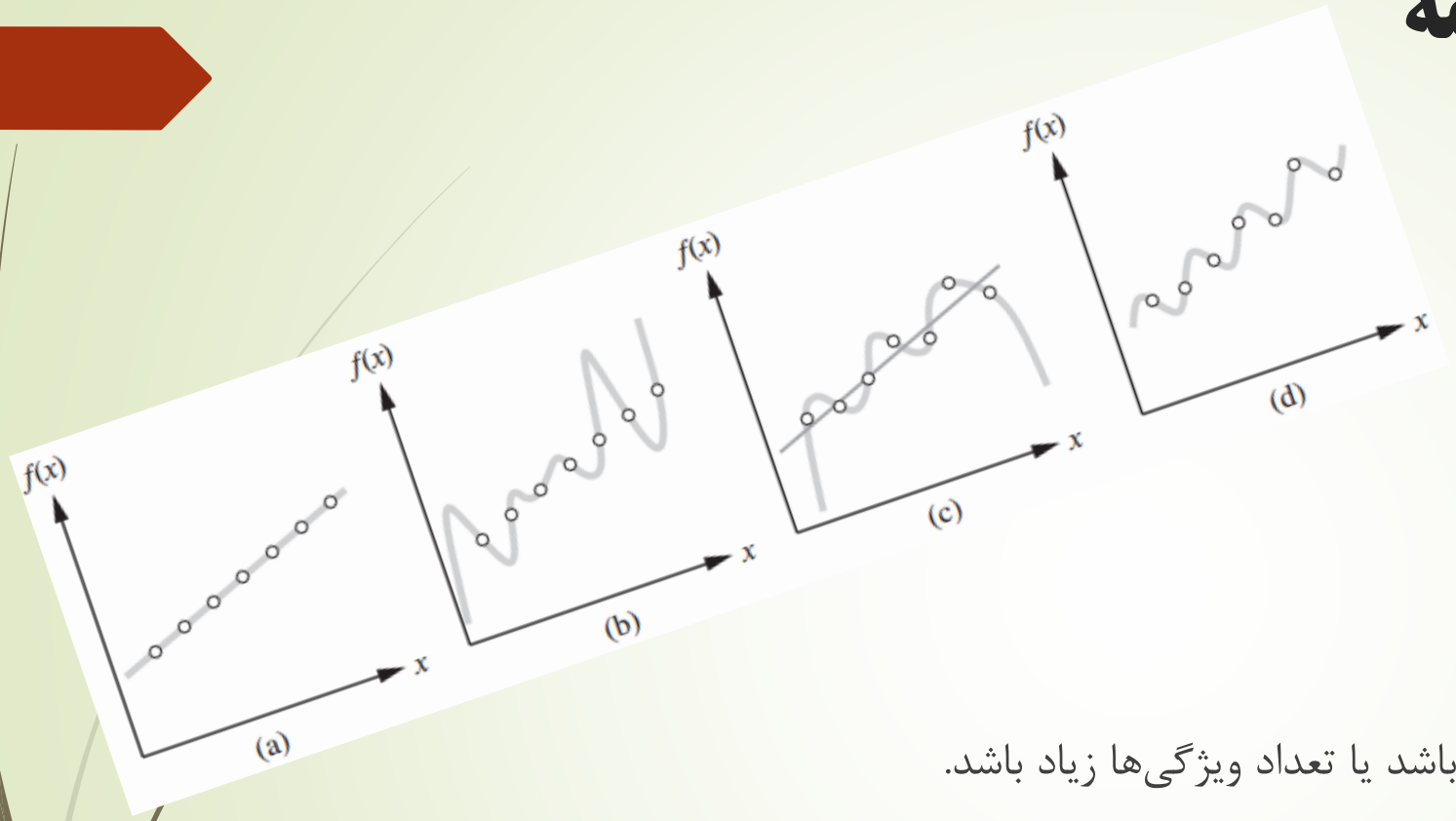
$$Gain(Patrons) = 1 - \left[\frac{2}{12}B\left(\frac{0}{2}\right) + \frac{4}{12}B\left(\frac{4}{4}\right) + \frac{6}{12}B\left(\frac{2}{6}\right)\right] \approx 0.541 \text{ bits},$$

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

تعمیم و بیش‌برازش

- اگر الگوی مشخصی برای یادگیری وجود نداشته باشد، الگوریتم درخت تصمیم ممکن است درختی بزرگ ارائه کند (و بی فایده).
- مثلا می‌خواهیم بدانیم یک طاس ۶ تایی سالم، ۶ می‌شود یا خیر؟ (متغیر دودوئی)
- نمونه‌ها = تعدادی طاس سالم به همراه نتیجه‌ای که با یک بار انداختن آنها مشاهده شده است.
- ویژگی‌ها: رنگ، وزن، زمان انداختن طاس، گرسنه بودن/نبودن فرد طاس‌انداز(!) و ...
- درخت تصمیم درست باید تنها دارای یک گره با مقدار «خیر» باشد.
- اما درختی که الگوریتم درخت تصمیم می‌دهد، همه ویژگی‌ها را به ترتیبی که از روی نمونه‌های موجود محاسبه می‌کند، آزمایش می‌کند و درختی بزرگ می‌سازد.
  - مثلا ممکن است اتفاقا در داده‌ها، دو طاس ۷گرمی آبی خروجی ۶ داشته باشند و درخت تصمیم ویژگی‌های وزن و رنگ را مهم بداند و ... .
  - چنین پدیده‌ای، بیش برازش یا Overfitting نام دارد.

تعمیم و بیش‌برازش... ادامه



- در شکل، منحنی b و c روی داده‌ها بیش برازش داشته‌اند.
- احتمال بیش‌برازش:
  - افزایش می‌یابد اگر فضای فرضیه بزرگ باشد یا تعداد ویژگی‌ها زیاد باشد.
  - کاهش می‌یابد اگر تعداد نمونه‌های آموزشی افزایش یابد.
- برای مقابله با بیش برازش در درخت تصمیم، درخت باید پس از ایجاد شدن، **هرس** شود.

هرس درخت تصمیم

- هرس درخت:
  - 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، زیرشاخه‌هایی ایجاد کند که نسبت نمونه مثبت و منفی در آنها تقریبا مانند همین گره باشد (یعنی p/(p+n))، آنگاه این ویژگی بی‌ربط است
    - 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؟)**

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

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

- هرس درخت باعث می‌شود نویز در داده‌ها تاثیر منفی کمتری داشته باشد.
  - ویژگی دارای نویز، دست‌آورد اطلاعات کمی دارد (نمونه‌ها را تصادفی به زیرشاخه‌ها تقسیم می‌کند)
- درخت هرس شده کوچک تر و قابل فهم تر است.

هرس یا خاتمه زود هنگام؟

- چرا به جای هرس، در حین ساخت درخت، وقتی ویژگی دارای دست‌آورد اطلاعات بالا پیدا نکردیم، ساخت زیرشاخه‌ها را متوقف نکنیم؟ (هزینه ساخت کمتر)
- ممکن است یک ویژگی خوب پیدا نشود، اما ترکیب چند ویژگی با هم دارای دست‌آورد 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، دست‌آورد اطلاعات صفر خواهد داشت
  - اما در عمق دوم درخت، انتخاب ویژگی بعدی، دست‌آورد اطلاعات برابر ۱ دارد.