دانشگاه آزاد اسلامی واحد تبریز

نام درس: یادگیری ماشین

بخش: یادگیری استقرایی و درخت‌های تصمیم

نام استاد: دکتر مسعود کارگر

فهرست مطالب

- یادگیری استقرایی
- درخت تصمیم
- قابلیت توصیف
- فضاهای فرضیه
- الگوریتم یادگیری درخت تصمیم
- انتخاب صفت
- استفاده از تئوری اطلاعات
- بهره اطلاعات
- اندازه‌گیری کارایی
- تطبیق بیش از حد و مقابله با آن

Inductive learning یادگیری استقرایی

- یک شیوه یادگیری با ناظر است.
- ساده‌ترین شکل یادگیری: یادگیری یک تابع از روی چند مثال

f تابع هدف مورد نظر است،

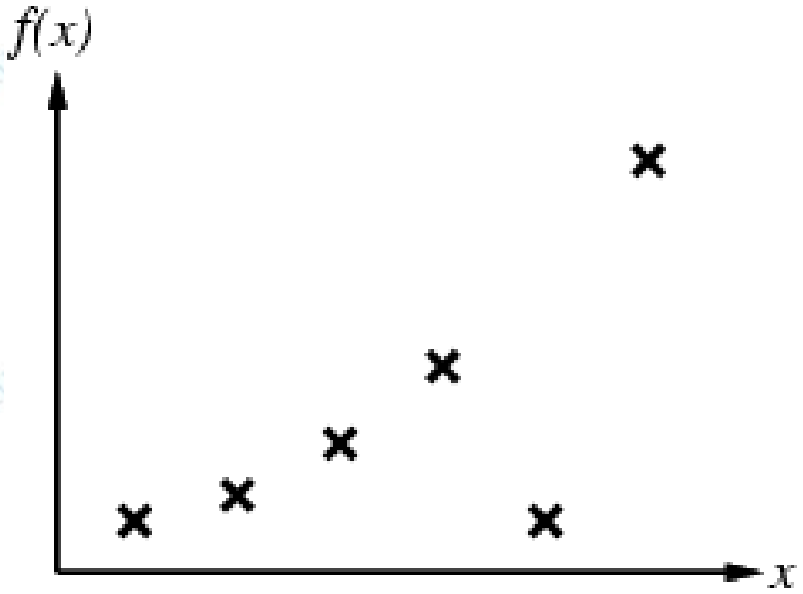
یک مثال از این تابع زوج مرتبی به شکل $(x, f(x))$ می‌باشد.

مساله ما یافتن تابع فرضیه h است به نحوی که:

- که تا حد ممکن به تابع اصلی منطبق باشد: $h \approx f$
- ورودی ما یک مجموعه آموزشی (training set) حاوی چندین مثال می‌باشد.
-

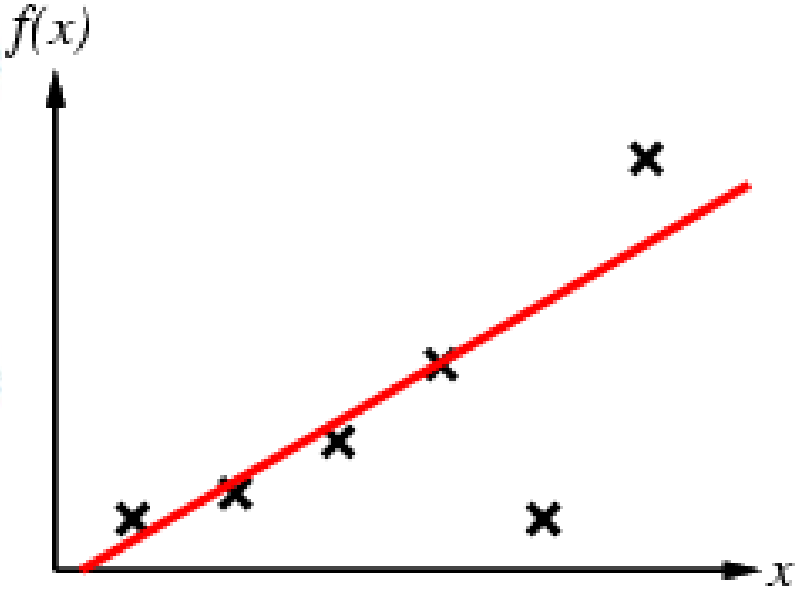
روش یادگیری استقرایی

- ایجاد و یا تنظیم تابع ***h*** به نحوی که روی مجموعه آموزشی با ***f*** تطابق داشته باشد.
- تابع ***h*** یک تابع سازگار **(consistent)** خواهد بود اگر با ***f*** روی تمام مثالها تطابق داشته باشد.
- مشابه روش برازاندن منحنی **(Curve fitting)**



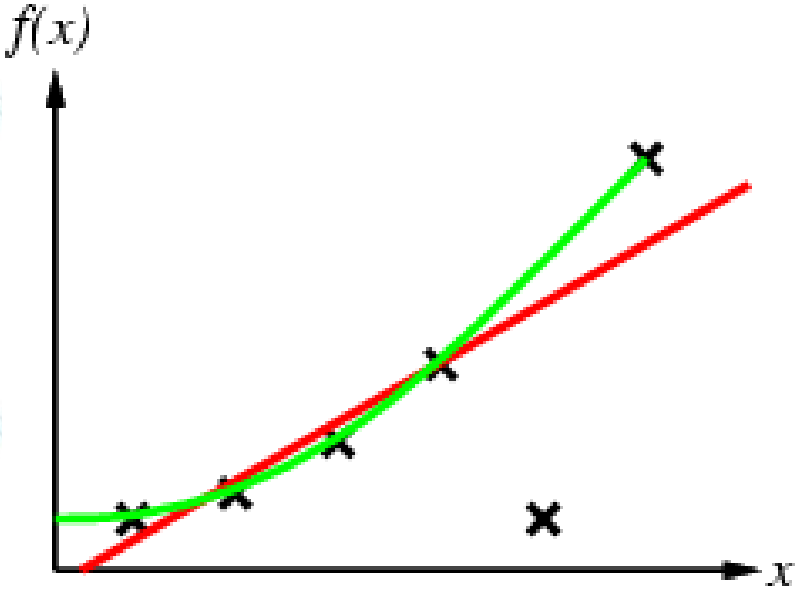
روش یادگیری استقرایی

- ایجاد و یا تنظیم تابع ***h*** به نحوی که روی مجموعه آموزشی با ***f*** تطابق داشته باشد.
- تابع ***h*** یک تابع سازگار **(consistent)** خواهد بود اگر با ***f*** روی تمام مثالها تطابق داشته باشد.
- مشابه روش برازاندن منحنی **(Curve fitting)**



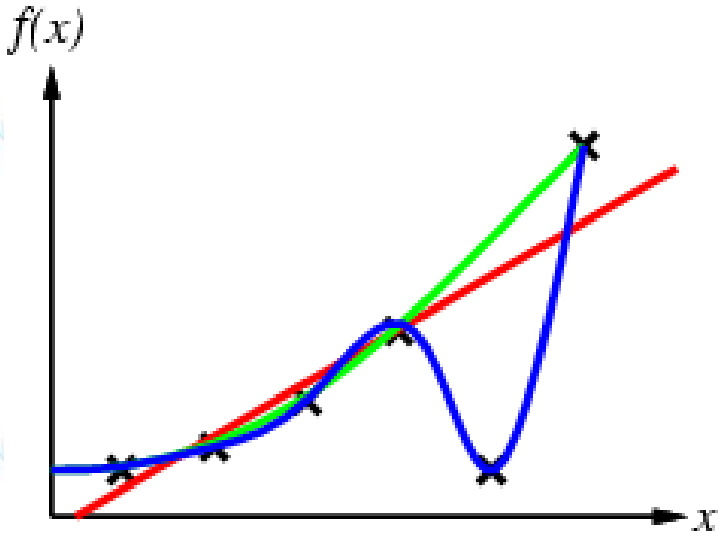
روش یادگیری استقرایی

- ایجاد و یا تنظیم تابع ***h*** به نحوی که روی مجموعه آموزشی با ***f*** تطابق داشته باشد.
- تابع ***h*** یک تابع سازگار **(consistent)** خواهد بود اگر با ***f*** روی تمام مثالها تطابق داشته باشد.
- مشابه روش برازاندن منحنی **(Curve fitting)**



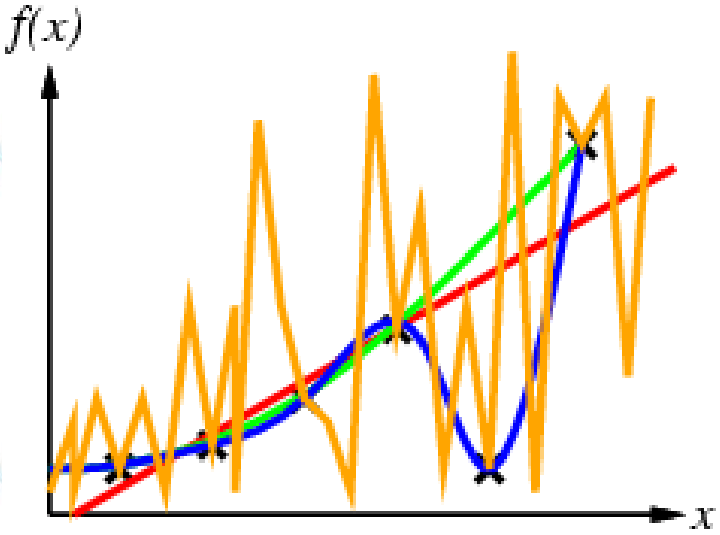
روش یادگیری استقرایی

- ایجاد و یا تنظیم تابع ***h*** به نحوی که روی مجموعه آموزشی با ***f*** تطابق داشته باشد.
- تابع ***h*** یک تابع سازگار (consistent) خواهد بود اگر با ***f*** روی تمام مثالها تطابق داشته باشد.
- مشابه روش برازاندن منحنی (Curve fitting)



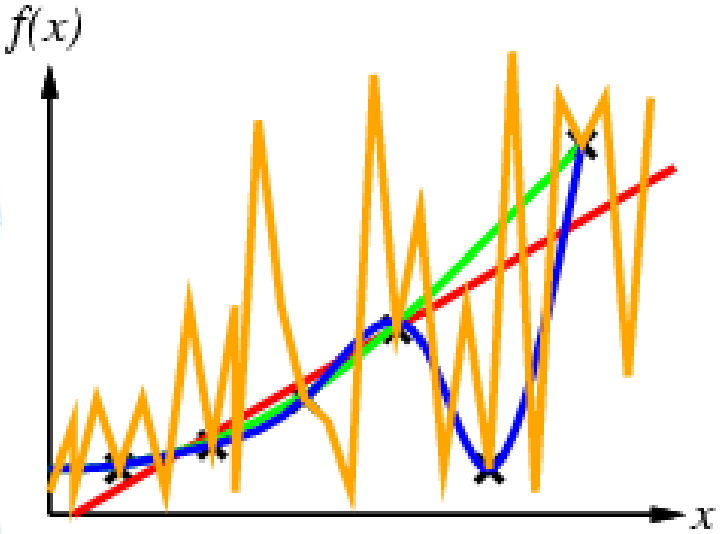
روش یادگیری استقرایی

- ایجاد و یا تنظیم تابع ***h*** به نحوی که روی مجموعه آموزشی با ***f*** تطابق داشته باشد.
- تابع ***h*** یک تابع سازگار **(consistent)** خواهد بود اگر با ***f*** روی تمام مثالها تطابق داشته باشد.
- مشابه روش برازاندن منحنی **(curve fitting)**



روش یادگیری استقرایی

- ایجاد و یا تنظیم تابع ***h*** به نحوی که روی مجموعه آموزشی با ***f*** تطابق داشته باشد.
- تابع ***h*** یک تابع سازگار **(consistent)** خواهد بود اگر با ***f*** روی تمام مثالها تطابق داشته باشد.
- مشابه روش برازاندن منحنی **(curve fitting)**



- اصل **Ockham**: ساده‌ترین فرضیه سازگار مورد ترجیح است.

Decision trees

یادگیری با درخت تصمیم

مساله: تصمیم‌گیری در مورد صبر کردن یا نکردن برای خالی شدن یک میز در یک رستوران با توجه به شرایط روی صفات زیر:

1. **Alternate**: آیا رستوران جایگزینی در نزدیکی این رستوران وجود دارد؟
2. **Bar**: آیا محلی برای صرف نوشیدنی در زمان انتظار در رستوران وجود دارد؟
3. **Fri/Sat**: آیا امروز یک روز آخر هفته است؟
4. **Hungry**: میزان گرسنگی ما چقدر است؟
5. **Patrons**: تعداد مشتریان موجود در رستوران چندتاست؟ **(None, Some, Full)**
6. **Price**: محدوده قیمت رستوران چیست؟ **($, $$, $$$)**
7. **Raining**: آیا باران در خارج رستوران می‌بارد؟
8. **Reservation**: آیا از قبل رزرو انجام داده‌ایم یا خیر؟
9. **Type**: رستوران از چه نوعی است؟ **(French, Italian, Thai, Burger)**
10. **WaitEstimate**: زمان تخمینی انتظار چقدر است؟ **(0-10, 10-30, 30-60, >60)**

نمایشهای مبتنی بر صفت
Attribute-based representations

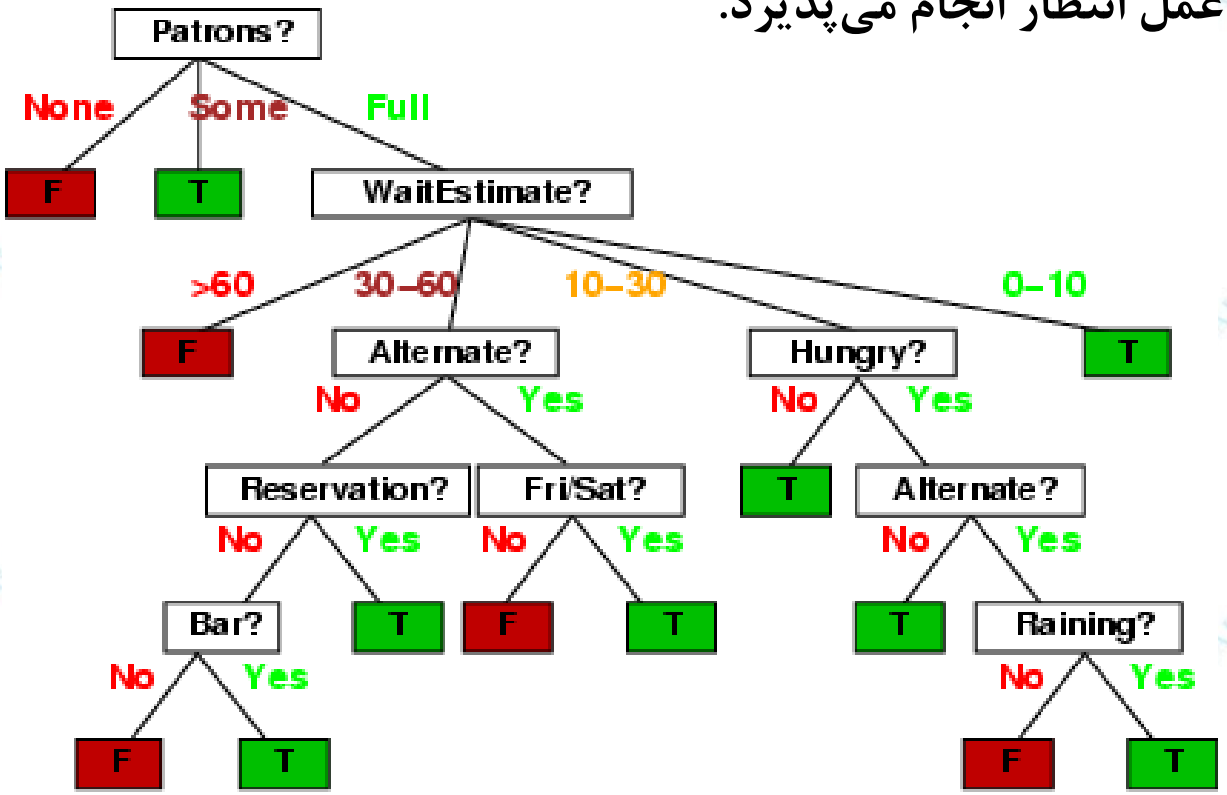
- **مثالها بوسیله مقادیر صفات (بولی، گسسته، پیوسته) توصیف می‌شوند.**

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

- **مثالها با مثبت (T) و منفی (F) طبقه‌بندی شده‌اند.**

Decision trees درخت تصمیم

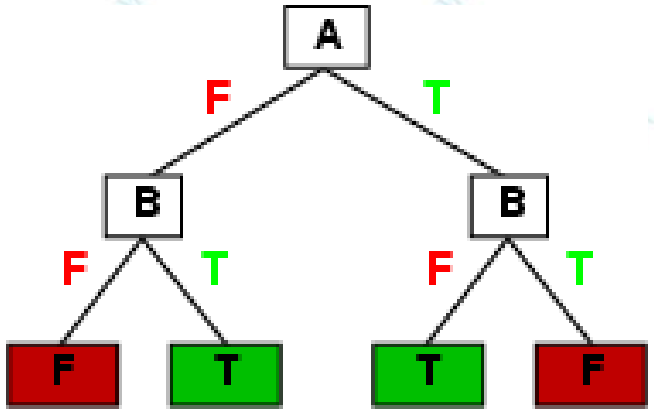
- یک شیوه ممکن برای نمایش فرضیه می‌باشد.
- در موارد **true** عمل انتظار انجام می‌پذیرد.



Expressiveness قابلیت توصیف

- درختهای تصمیم قابلیت توصیف هر تابعی از صفات ورودی را دارا می‌باشند.
  - به عبارت دیگر هر تابع بولی بیان شده با یک جدول حقیقت توسط درخت تصمیم قابل توصیف است. در این حال هر سطر جدول یک مسیر از ریشه تا برگ است.

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



- بنابراین برای هر مجموعه آموزشی یک درخت تصمیم سازگار وجود دارد، (مگر اینکه تابع f غیر قطعی باشد) اما این احتمال وجود دارد که مثالهای جدید در درخت قابل تعمیم نباشند.

- ترجیح با درختهای تصمیم فشرده است.

Hypothesis spaces فضاهای فرضیه

سوال: چند درخت تصمیم (فرضیه) متمایز برای n صفت بولی وجود دارد؟

= تعداد توابع بولی متمایز که می‌توان با n صفت بولی ساخت.

= تعداد جداول حقیقت متمایز با 2^n سطر = 2^{2^n}.

- بطور مثال برای ۶ صفت بولی ۱۸،۴۴۶،۷۴۴،۰۷۳،۷۰۹،۵۵۱،۶۱۶ درخت وجود دارد.

Hypothesis spaces فضاهای فرضیه

سوال: چند درخت تصمیم (فرضیه) متمایز برای n صفت بولی وجود دارد؟

= تعداد توابع بولی متمایز که می‌توان با n صفت بولی ساخت.

= تعداد جداول حقیقت متمایز با 2^n سطر = 2^{2^n}.

- بطور مثال برای ۶ صفت بولی ۱۸،۴۴۶،۷۴۴،۰۷۳،۷۰۹،۵۵۱،۶۱۶ درخت وجود دارد.

سوال: چند فرضیه عطفی خالص وجود دارد؟

- هر فرضیه عطفی (نظیر $Hungry \wedge \neg Rain$) یک مسیر از ریشه تا برگ در درخت تصمیم است.
- هر صفت ممکن است در یک فرضیه عطفی نقش مثبت، منفی یا بدون تاثیر داشته باشد.
  - در نتیجه برای n صفت، 3^n فرضیه عطفی متمایز وجود دارد.
- افزایش حجم فضای فرضیه باعث می‌شود:
  - شانس توصیف تابع هدف افزایش یابد.
  - از طرف دیگر باعث افزایش تعداد فرضیه‌های سازگار با مجموعه آموزشی می‌شود و در نتیجه احتمال پیش بینی غاط نیز بیشتر می‌شود.

پس نیاز به یک الگوریتم هوشمند داریم تا در این فضا یک درخت سازگار بیابد.

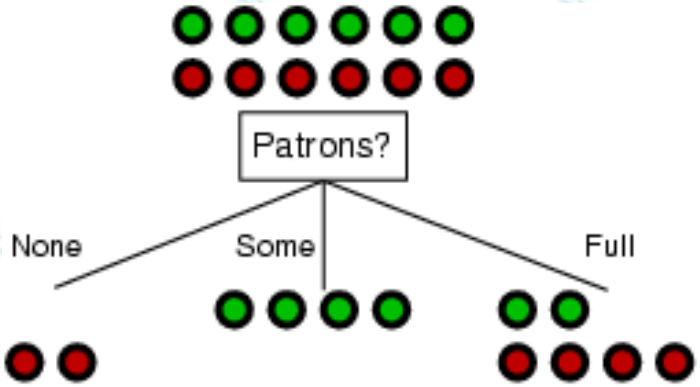
الگوریتم یادگیری درخت تصمیم

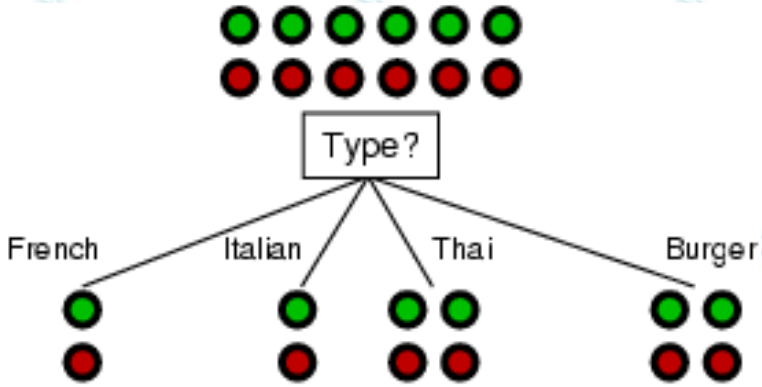
- هدف: یافتن کوچکترین درخت سازگار با مثالهای آموزشی است.
- ایده: انتخاب (بازگشتی) با معنی‌ترین صفت برای ریشه (زیر) درخت.

```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value v_i of best do
            examples_i ← {elements of examples with best = v_i}
            subtree ← DTL(examples_i, attributes − best, MODE(examples))
            add a branch to tree with label v_i and subtree subtree
        return tree
```

انتخاب صفت

- ایده: یک صفت خوب در حالت ایده‌آل مثالها را به دو زیرمجموعه تمام مثبت و تمام منفی تفکیک می‌کند.





- بنابراین *Patrons?* انتخاب بهتری است.

استفاده از تئوری اطلاعات

- در پیاده‌سازی `Choose-Attribute` در الگوریتم DTL از محتوای اطلاعاتی (انتروپی) استفاده می‌شود.

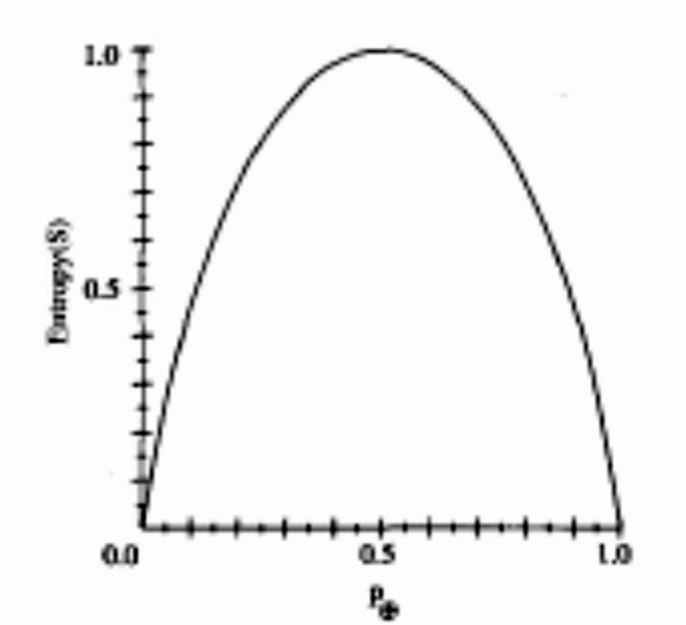
$$I(P(v_1), \ldots, P(v_n)) = \sum_{i=1}^{n} -P(v_i) \log_2 P(v_i)$$

- برای یک مجموعه آموزشی حاوی p مثال مثبت و n مثال منفی:

$$I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

استفاده از تئوری اطلاعات

- در حالتیکه دو کلاس مثبت و منفی داریم:
  - اگر $I=0$ باشد یعنی همه مثالها در یک کلاس هستند.
  - اگر $I=1$ باشد یعنی نصف مثالها در یک کلاس و نصف دیگر در کلاس دوم هستند.



Information gain بهره اطلاعات

- یک صفت منتخب نظیر A مجموعه آموزشی E را به زیرمجموعه‌های $E_1, \ldots, E_v$ بر اساس مقادیر A تقسیم می‌نماید. با فرض آنکه A دارای v مقدار متمایز است. میزان اطلاعات صفت A عبارت است از:

$$remainder(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p+n} I\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$$

- بهره اطلاعات (IG) که حاصل تفریق انتروپی و میزان اطلاعات صفت A است معیار مناسبی برای انتخاب صفت است.

$$IG(A) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - remainder(A)$$

- صفتی انتخاب می‌شود که بیشترین مقدار بهره اطلاعات (IG) را داشته باشد.

Information gain بهره اطلاعات

برای مجموعه آموزشی ذکر شده داریم: $p = n = 6,\ I(6/12, 6/12) = 1$ bit

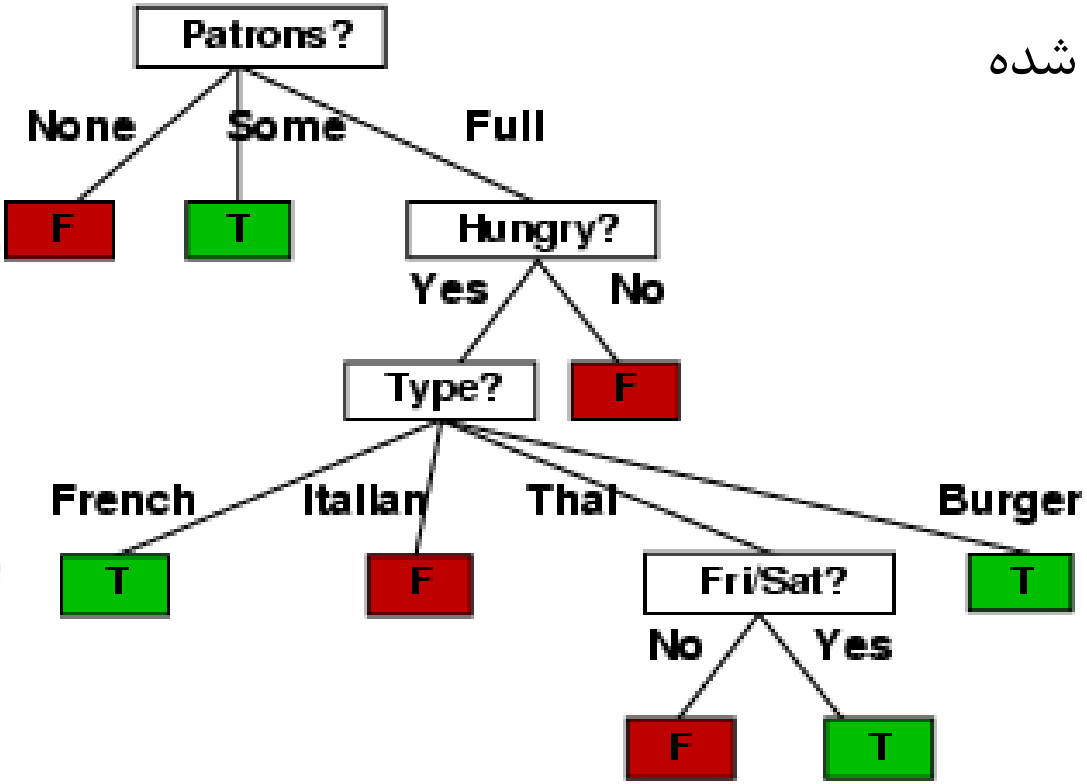
برای این مجموعه صفات *Patrons* و *Type* را در نظر بگیرید (و همچنین سایر صفات):

$$IG(Patrons) = 1 - [\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I(\frac{2}{6}, \frac{4}{6})] = .0541 \text{bits}$$

$$IG(Type) = 1 - [\frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{2}{12} I(\frac{1}{2}, \frac{1}{2}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4}) + \frac{4}{12} I(\frac{2}{4}, \frac{2}{4})] = 0 \text{ bits}$$

در این مجموعه صفت *Patrons* بیشترین *IG* را نسبت به سایر صفات داراست پس به عنوان ریشه درخت توسط الگوریتم DTL انتخاب می‌شود

مثال



- درخت تصمیم آموزش داده شده با ۱۲ مثال:

- این درخت ساده‌تر از درخت فرضیه واقعی است. اساسا مجموعه‌های آموزشی کوچک فرضیه‌های پیچیده ایجاد نمی‌کند.

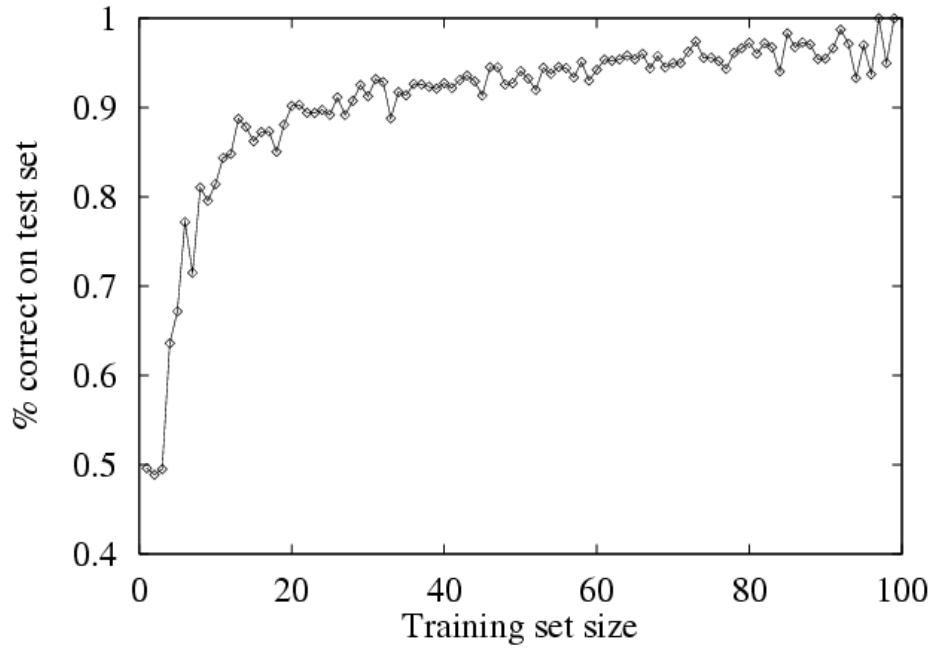
اندازه‌گیری کارایی
Performance measurement

- چگونه می‌توان مطمئن شد که $h \approx f$؟

1. استفاده از قضایای محاسباتی/ آماری تئوری اطلاعات.
2. آزمایش h روی یک مجموعه آزمایشی جدید.

(امتحان الگوریتم روی اندازه‌های متفاوت از مجموعه آموزشی و تست روی باقیمانده داده‌ها به عنوان مجموعه آزمایشی)

منحنی آموزش = درصد درستی پاسخ روی مجموعه آزمایشی نسبت به اندازه مجموعه آموزشی



ID3 Decision tree learning

ID3(*Examples*, *Target_attribute*, *Attributes*)

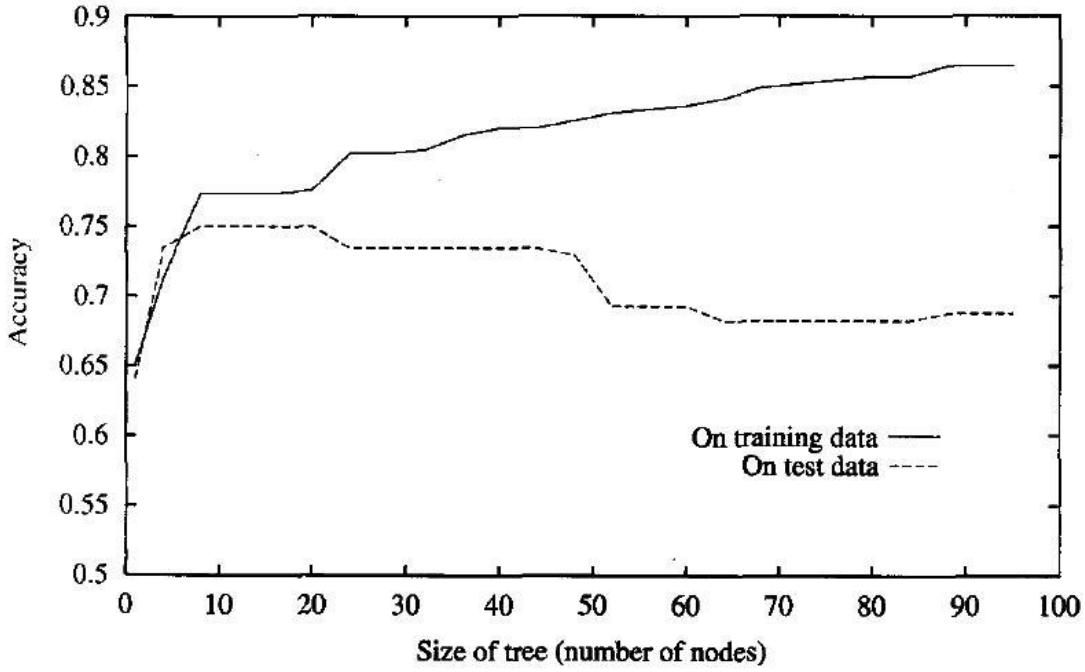
Examples are the training examples. Target_attribute is the attribute whose value is to be predicted by the tree. Attributes is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given Examples.

- Create a *Root* node for the tree
- If all *Examples* are positive, Return the single-node tree *Root*, with label = +
- If all *Examples* are negative, Return the single-node tree *Root*, with label = −
- If *Attributes* is empty, Return the single-node tree *Root*, with label = most common value of *Target_attribute* in *Examples*
- Otherwise Begin
  - $A \leftarrow$ the attribute from *Attributes* that best* classifies *Examples*
  - The decision attribute for *Root* $\leftarrow A$
  - For each possible value, v_i, of A,
    - Add a new tree branch below *Root*, corresponding to the test $A = v_i$
    - Let $Examples_{v_i}$ be the subset of *Examples* that have value v_i for A
    - If $Examples_{v_i}$ is empty
      - Then below this new branch add a leaf node with label = most common value of *Target_attribute* in *Examples*
      - Else below this new branch add the subtree ID3($Examples_{v_i}$, *Target_attribute*, *Attributes* − {A}))
- End
- Return *Root*

* The best attribute is the one with highest *information gain*, as defined in Equation (3.4).

تطبیق بیش از حد (Over fitting)

- گاهی اوقات در مجموعه آموزشی مثالهای نادری وجود دارد که ممکن است با توزیع کلی داده‌ها مطابقت نداشته باشند. تعداد زیاد گره‌های یک درخت تصمیم باعث می‌شود تا درخت درجه آزادی زیادی برای انطباق با این مثالها داشته باشد.



مقابله با تطبیق بیش از حد

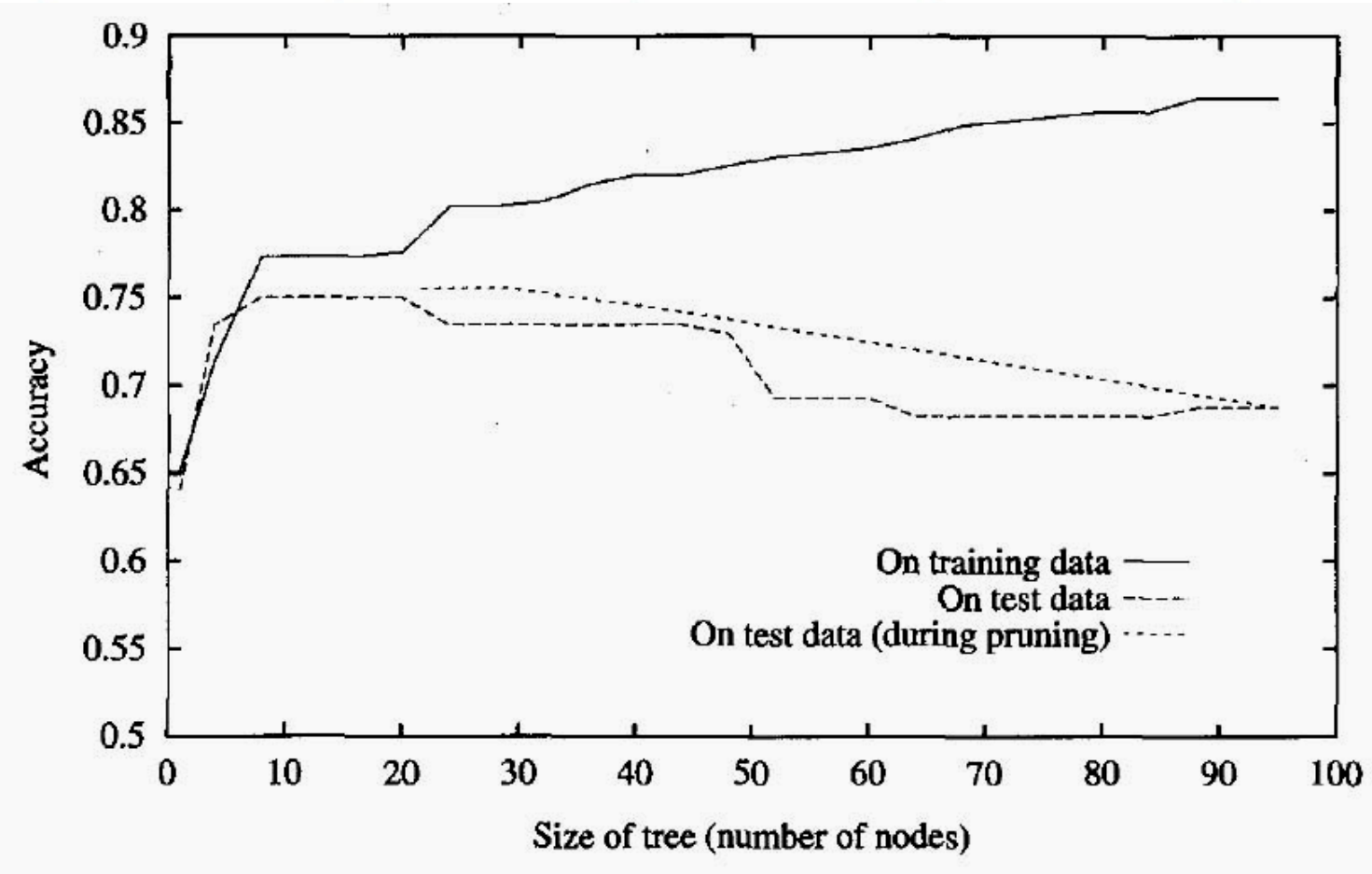
- اجتناب از رشد درخت در مرحله ایجاد.
- هرس کردن (Post-Pruning) درخت پس از ایجاد.

- روشهای تعیین اندازه درخت:
  - استفاده از دو مجموعه جداگانه آموزش و تست جهت ارزیابی.
  - بکارگیری تستهای آماری (نظیر روش Quinlan 86) برای تخمین بسط دادن یا هرس کردن یک گره.
  - استفاده از توابع هیوریستیک برای تشخیص اندازه مناسب درخت. (نظیر روش Quinlan & Rivest 89 و روش Mehta et al.95)

هرس خطای کاهش‌یافته
Reduced Error Pruning

- یک روش مقابله با Over fitting.
- ابداع شده توسط Quinlan 1987.
  - حذف زیردرخت و جایگزینی آن با ریشه زیردرخت.
  - تخصیص کلاس اکثریت به گره جایگزین شده.
  - حذف زیردرخت، مشروط به اینکه خطای گره جایگزین شده بیشتر از حالت اولیه نباشد.

هرس خطای کاهش‌یافته
Reduced Error Pruning



قدردانی

- **دکتر شیری:**
  - **از دانشگاه صنعتی امیرکبیر**