Machine Learning

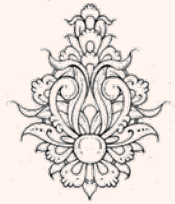
Decision Tree

یادگیری ماشین
(۱۳-۱۱-۸۰۵-۰۱)
فصل نهم

دانشگاه شهید بهشتی
دانشکده‌ی مهندسی برق و کامپیوتر
پاییز ۱۳۹۳
احمد محمودی ازناوه

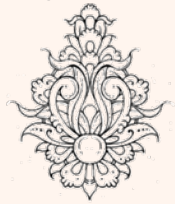
فهرست مطالب

- درخت تصمیم
- درخت تصمیم تک‌متغیره
  - درخت‌های دسته‌بندی
  - درخت‌های رگرسیون
- هرس کردن
- استخراج قانون با کمک درخت
- استنتاج قانون
- درخت‌های چند متغیره



یادگیری درخت تصمیم

- در روش‌های پارامتری برای همه‌ی داده‌ها یک مدل به دست می‌آید، در حالی که در روش‌های ناپارامتری داده‌ها بر اساس یک معیار فاصله به «نواحی محلی» تقسیم می‌شوند، سپس برای هر ناحیه یک مدل استخراج می‌شود.
  - در این حالت یافتن ناحیه‌ی مربوط هزینه‌بر است؛ به ازای هر ورودی همه‌ی داده‌های آموزشی جستجو شده تا ناحیه‌ی مورد نظر پیدا شود.
- «درخت تصمیم» یک ساختمان‌داده‌ی سلسله‌مراتبی است که برای یادگیری «باناظر» مورد استفاده قرار می‌گیرد.
  - با تعداد مراحل کم‌تری ناحیه‌ی محلی شناسایی می‌شود.
  - بر پایه‌ی سیاست «تفرقه‌بینداز و حکومت‌کن» عمل می‌کند.
  - این شیوه نیز «ناپارامتری» است، از این حیث که یک مدل برای داده‌ها به دست نمی‌آورد.
  - بیشتر در داده‌کاوی مورد استفاده قرار می‌گیرد.



ساختار درخت تصمیم

- درخت تصمیم از یک سری گره‌ی «تصمیم (داخلی)» و «برگ» تشکیل شده است.

internal decision node

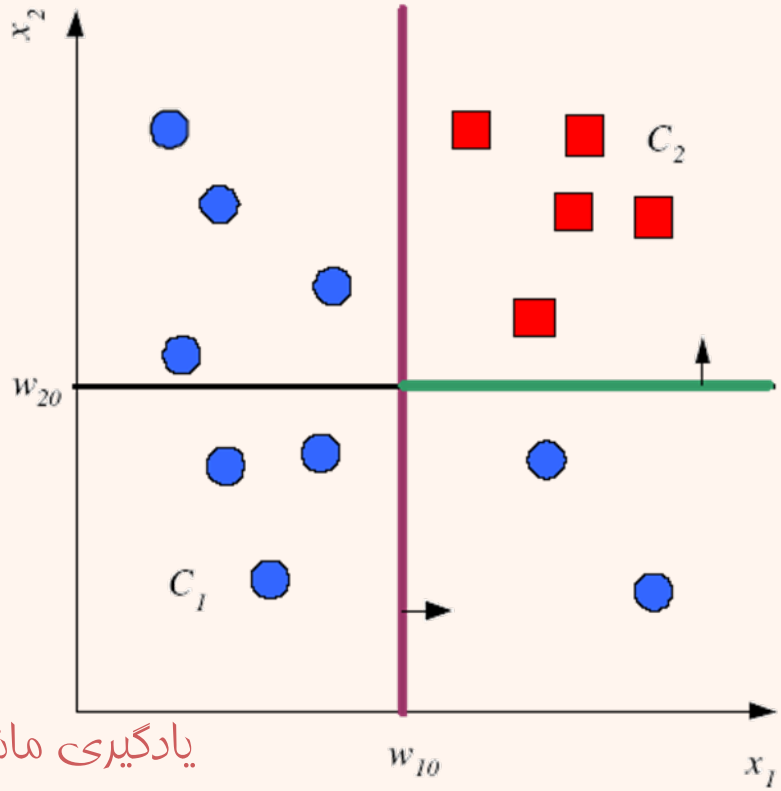
- گره‌های تصمیم، معادل یک «تابع آزمون($f_m(x)$)» هستند که بر اساس نتیجه‌ی آن‌ها خروجی‌ها برچسب می‌خورند. به ازای هر ورودی جستجو از ریشه شروع شده و در هر گره‌ی تصمیم یکی از انشعاب‌ها انتخاب می‌شود. این فرآیند تا رسیدن به برگ ادامه می‌یابد.

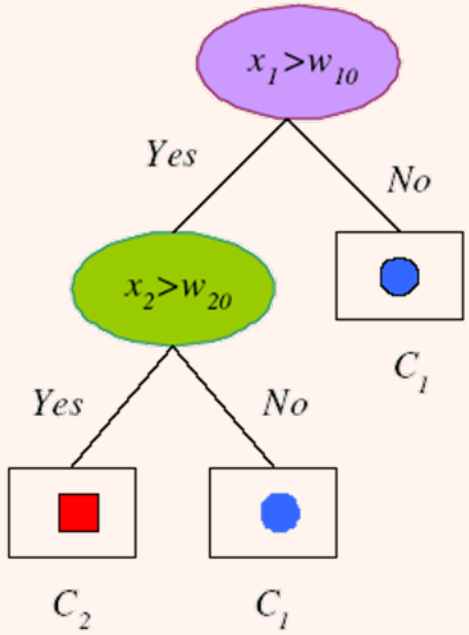
Leaves

- مقدار گره‌ی برگ خروجی به ازای آن ورودی را نشان می‌دهد. در دسته‌بندی شماره‌ی کلاس و در رگرسیون مقدار میانگین

ساختار درخت تصمیم(ادامه...)

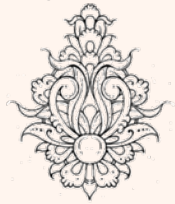
- هر «تابع آزمون($f_m(\boldsymbol{x})$)» یک جداساز در فضای d-بعدی است، که فضای ورودی را به نواحی کوچک‌تر تقسیم می‌کند. هر f یک تابع ساده است که با ترکیب با یکدیگر توابعی پیچیده را خواهند ساخت.





ساختار درخت تصمیم(ادامه...)

- بدین‌ترتیب درخت تصمیم، به یافتن نمونه‌های مشابه سرعت می‌بخشد. به عنوان مثال در صورت داشتن b ناحیه‌ی مختلف و در صورتی که تصمیم‌ها دودویی باشند، در بهترین حالت $\log_2 b$ گام تا رسیدن به ناحیه‌ی محلی نیاز است.
- از دیگر مزایای درخت تصمیم «قابلیت تفسیرپذیری» آن است.
  - می‌توان از درخت تشکیل شده یک‌سری قانون به صورت if-then استخراج کرد.



Univariate Trees

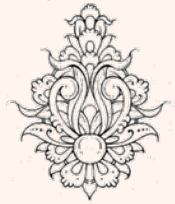
- در «درخت‌های تک‌متغیره»، در هر گره‌ی داخلی تنها یکی از ابعاد ورودی مورد بررسی قرار می‌گیرد.
  - در صورتی که متغیرها گسسته باشند، مقدار آن‌ها (با یکی از n مقدار ممکن) بررسی می‌شود.
  - و گرنه مقایسه‌ای صورت می‌گیرد که نتیجه‌ی گسسته داشته باشد:
    - $f_m(\boldsymbol{x}) : x_j \geq w_{m0}$
  - بدین‌ترتیب فضای ورودی را به دو قسمت تقسیم می‌شود:

$$L_m = \{\mathbf{x} \mid x_j \geq w_{m0}\} \qquad R_m = \{\mathbf{x} \mid x_j < w_{m0}\}$$

Binary Split

آموزش درخت تصمیم

- به فرآیند ساخت درخت بر اساس داده‌های آموزشی «tree induction» می‌گویند.
- برای یک مجموعه‌ی آموزشی درخت‌های زیادی می‌توان در نظر گرفت.
- مطلوب‌ترین درخت، کوچک‌ترین درخت از نظر تعداد گره‌ها و سادگی عملیات مقایسه است.
- بیشتر الگوریتم‌های یادگیری درخت تصمیم حریصانه است.
  - در هر گام در پی بهترین جداسازی است.



(Breiman et al, 1984; Quinlan, 1986, 1993)

درخت‌های دسته‌بندی

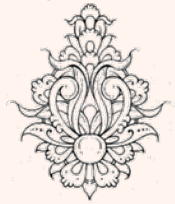
Impurity measure

- «ناخالصی» معیاری برای تشخیص میزان مطلوبیت درخت دسته‌بندی است.
- یک انشعاب (split) «خالص» است، در صورتی که هر بخش آن شامل نمونه‌های یک کلاس باشد.
  - N_m – تعداد نمونه‌هایی است که به گره‌ی m-ام رسیده‌اند (کل گره‌ها N است).
  - از بین آن‌ها N^i_m متعلق به کلاس i-ام(C_i) است.

$$\sum_i N_m^i = N_m$$

- احتمال این که نمونه‌ای که به گره‌ی m رسیده است، به دسته‌ی i-ام تعلق داشته باشد:

$$\hat{P}(C_i \mid \mathbf{x}, m) \equiv p_m^i = \frac{N_m^i}{N_m}$$

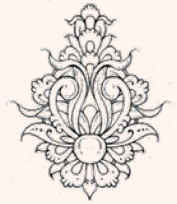


درخت‌های دسته‌بندی (ادامه...)

- گره‌ی m خالص است، اگر

$$p_m^i = (0 \text{ or } 1) \quad \text{for all } i$$

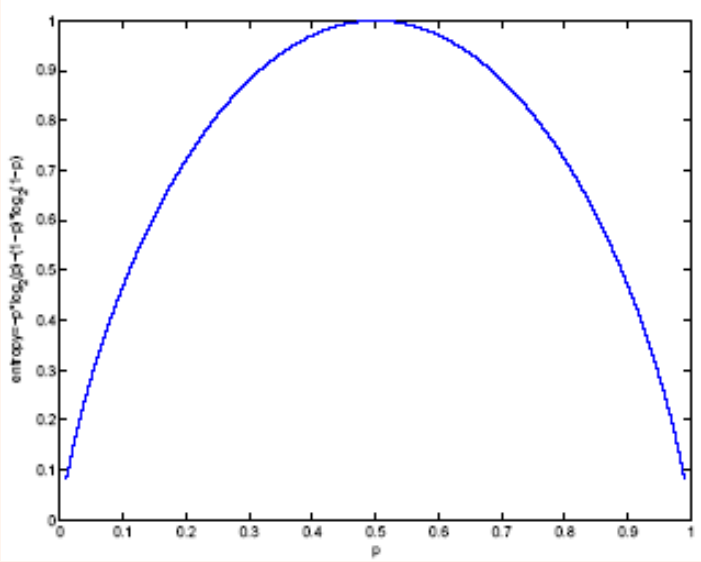
- در صورتی که همه‌ی جداسازی‌ها خالص باشند، لزومی به انشعاب مجدد درخت نیست و می‌توان برگ‌ها با برچسب دسته را به درخت افزود.
- فرآیند آموزش درخت باید به گونه‌ای صورت پذیرد که در هر گام میزان خلوص افزایش یابد.
- از این جهت لازم است معیاری برای سنجش خلوص در نظر گرفته شود.



خلوص

- یک انتخاب برای سنجش میزان خلوص، «آنتروپی اطلاعات» است.
  - آنتروپی اطلاعات: رخداد اتفاقی تا چه حد تصادفی است.

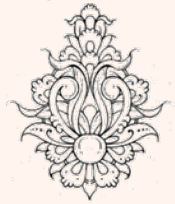
$$I_m = -\sum_{i=1}^{K} p_m^i \log_2 p_m^i$$



سایر معیارها

- آنتروپی تنها معیار مناسب نیست؛ برای دسته‌بندی دو گروه، هر تابع نامنفی $\phi(p,1-p)$ که دارای خصوصیات زیر باشد، به عنوان معیار خلوص قابل استفاده است.

  - $\phi(1/2,1/2) \geq \phi(p,1-p), \quad \text{for any } p \in [0,1]$
  - $\phi(0,1) = \phi(1,0) = 0$
  - $\phi(p,1-p)$ is increasing in p on $[0,1/2]$ and decreasing in p on $[1/2,1]$



Devroye, L., et al. (1996). A Probabilistic Theory of Pattern Recognition, Springer.

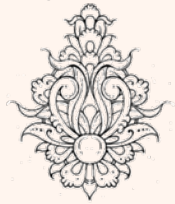
سایر معیارها (ادامه...)

$$entropy: \phi(p,1-p) = -p\log_2(p) - (1-p)\log_2(1-p)$$

$$Gini\ index: \phi(p,1-p) = 2p(1-p)$$

$$Missclassification\ error: \phi(p,1-p) = 1 - \max(p,1-p)$$

- این معیارها قابل تعمیم به K>2 هستند.
- پژوهش‌ها نشان داده است که این سه معیار در عمل تفاوت چندانی با هم ندارند.



تشکیل گره‌ی تصمیم

- انشعابی انتخاب می‌شود که باعث کاهش ناخالصی شود.
- در صورت که از N_m ورودی که به گره‌ی m–ام رسیده‌اند، N_{mj} ورودی انشعاب j–ام این گره را انتخاب کنند، خواهیم داشت:

$$\sum_{j=1}^{n} N_{mj} = N_m \qquad \sum_{i=1}^{K} N_{mj}^{i} = N_{mj} \qquad \sum_{j=1}^{n} N_{mj}^{i} = N_m^i$$

- بعد از انشعاب، و در صورت انتخاب انشعاب j –ام احتمال تعلق به کلاس i–ام:

$$\hat{P}(C_i \mid \mathbf{x}, m, j) \equiv p_{mj}^{i} = \frac{N_{mj}^{i}}{N_{mj}}$$

- و «ناخالصی کل» به صورت زیر به دست می‌آید:

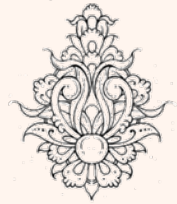
$$I'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p_{mj}^{i} \log_2 p_{mj}^{i}$$

آموزش درخت‌های دسته‌بندی

- برای همه‌ی خصیصه‌ها، ناخالصی به ازای تمام حالات محاسبه می‌شود:
  - در حالتی که خصیصه‌ها گسسته باشند(در صورتی که خصیصه‌ی مورد نظر دارای n مقدار مختلف باشد)، هر گره‌ی تصمیم به n گره‌ی دیگر، منشعب می‌شود.
  - در مورد خصیصه‌های پیوسته، هر گره به دو گره‌ی دیگر منشعب می‌شود.

$f_m(x) : x_j \geq w_{m0}$

    - در این حالت باید مقدار آستانه‌ی w_{m0} نیز مشخص شود، N_m-1 مقدار ممکن برای این حد آستانه وجود دارد، به ازای همه‌ی خصیصه‌ها و همه‌ی حد آستانه‌های ممکن ناخالصی محاسبه می‌شود.
- به جای ذخیره‌ی برچسب گروه در برگ‌ها، بهتر است احتمال تعلق به کلاس (posterior probability) ذخیره شود، چرا که ممکن است در مراحل بعدی، مانند محاسبه‌ی ریسک به کار آید.



```
GenerateTree(X)
    If NodeEntropy(X) < θ_I /* eq. 9.3
        Create leaf labelled by majority class in X
        Return
    i ← SplitAttribute(X)
    For each branch of x_i
        Find X_i falling in branch
        GenerateTree(X_i)
SplitAttribute(X)
    MinEnt ← MAX
    For all attributes i = 1, ..., d
        If x_i is discrete with n values
            Split X into X_1, ..., X_n by x_i
            e ← SplitEntropy(X_1, ..., X_n) /* eq. 9.8 */
            If e<MinEnt MinEnt ← e; bestf ← i
        Else /* x_i is numeric */
            For all possible splits
                Split X into X_1, X_2 on x_i
                e←SplitEntropy(X_1, X_2)
                If e<MinEnt MinEnt ← e; bestf ← i
    Return bestf
```

$$\mathcal{I}_m = -\sum_{i=1}^{K} p_m^i \log_2 p_m^i$$

در صورتی که تشکیل درخت تا حالتی که ناخالصی به صفر برسد ادامه یابد ممکن است منجر به overfitting شود(به ویژه با وجود نویز). در این حالت یک حدآستانه برای ناخالصی در نظر گرفته می‌شود. این معیار (θ_I) پیچیدگی مدل را مشخص می‌کند.

$$\mathcal{I}'_m = -\sum_{j=1}^{n} \frac{N_{mj}}{N_m} \sum_{i=1}^{K} p_{mj}^i \log_2 p_{mj}^i$$

درخت‌های رگرسیون

- درخت‌های رگرسیون، مانند درخت‌های دسته‌بندی هستند؛ با یک معیار ناخالصی مناسب:

«میانگین مربعات خطا»

$$b_m(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \in \mathcal{X}_m : \mathbf{x} \text{ reaches node } m \\ 0 & \text{otherwise} \end{cases}$$

مقدار تخمین زده شده در گره‌ی m-ام

$$E_m = \frac{1}{N_m} \sum_t \left(r^t - g_m\right)^2 b_m\left(\mathbf{x}^t\right)$$

- می‌توان از میانگین(میانه برای داده‌های نویزی) برای تخمین خروجی استفاده کرد.

$$g_m = \frac{\sum_t b_m\left(\mathbf{x}^t\right) r^t}{\sum_t b_m\left(\mathbf{x}^t\right)}$$

در صورت در نظر گرفتن میانگین داده‌های هر برگ، معیار خطا واریانس داده‌ها خواهد بود

درخت‌های رگرسیون(ادامه...)

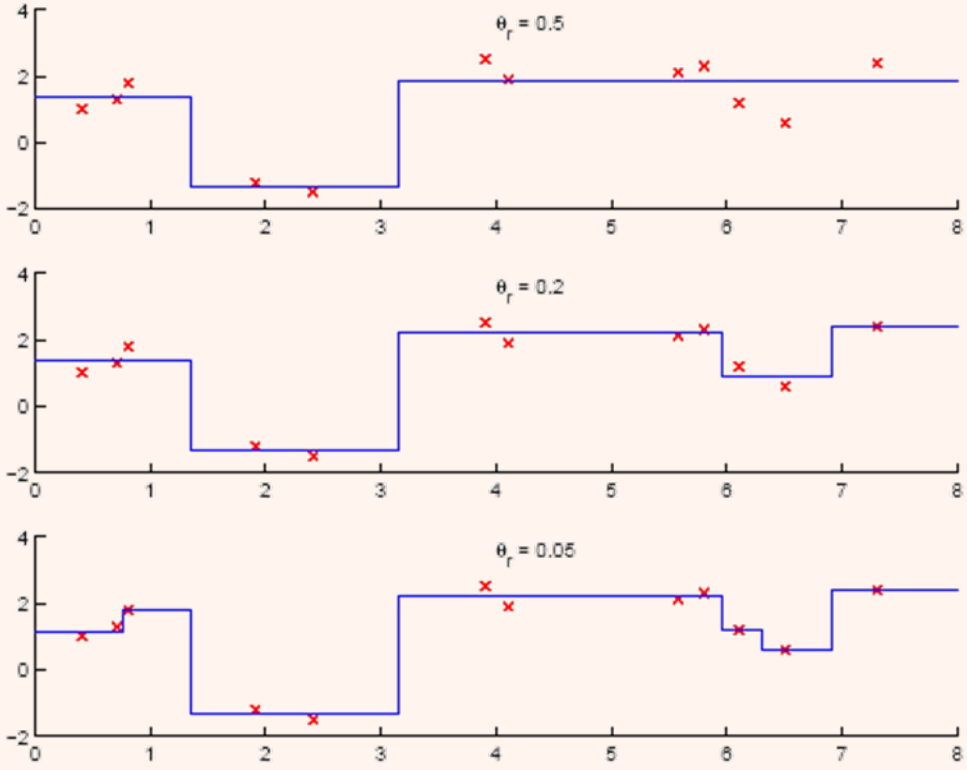
- در صورتی که میزان خطا قابل پذیرش باشد، گره‌ی برگ ایجاد می‌شود، وگرنه روند تقسیم ادامه می‌یابد.
- علاوه بر میانگین مربعات خطا، می‌توان از معیار «بدترین خطای ممکن» نیز استفاده کرد، بدین‌ترتیب می‌توان در بدترین حداکثر میزان خطا را محدود نمود:

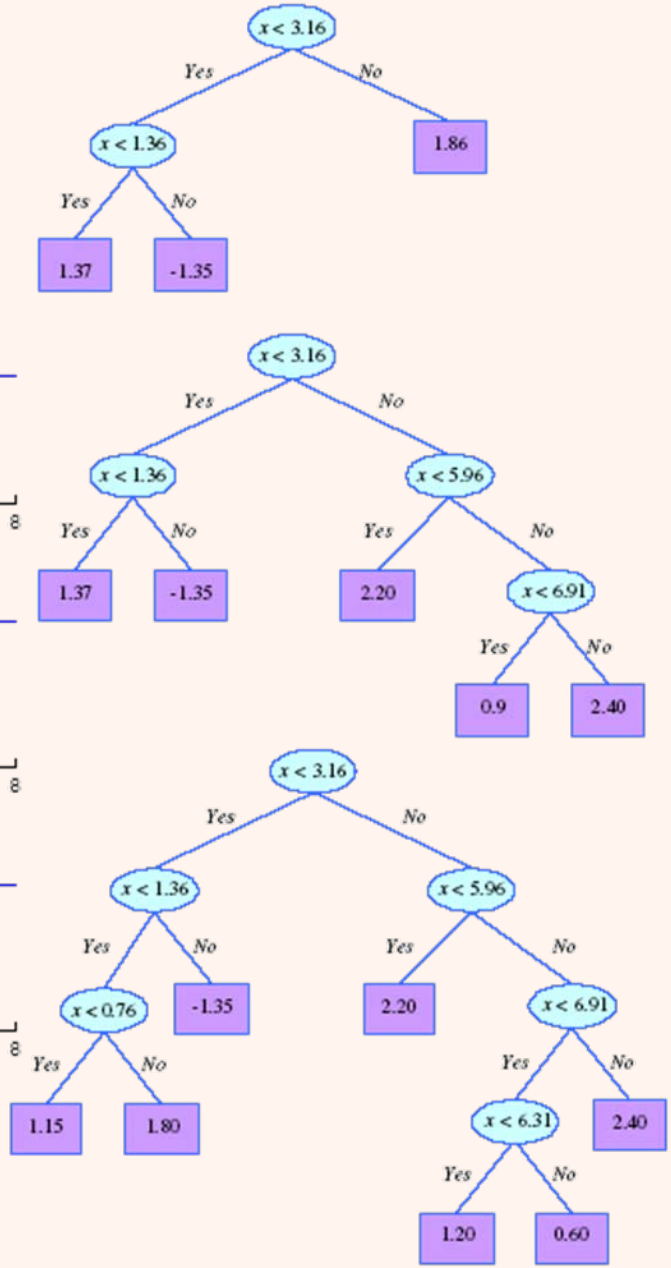
$$E_m = \max_j \max_t \left| r^t - g_{mj} \right| b_m\left(\mathbf{x}^t\right)$$

- میزان پیچیدگی مدل با حداکثر خطای قابل پذیرش(θ_r) مشخص می‌شود.
- به جای میانگین‌گیری، می‌توان در هر برگ برای داده‌ها یک مدل خطی نیز در نظر گرفت(رگرسیون خطی):

$$g_m\left(\mathbf{x}\right) = \mathbf{w}_m^T \mathbf{x} + w_{m0}$$

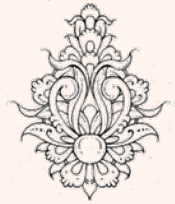
مثال


θr = 0.5
θr = 0.2
θr = 0.05


x < 3.16
Yes
No
x < 1.36
1.86
Yes
No
1.37
-1.35
x < 3.16
Yes
No
x < 1.36
x < 5.96
Yes
No
Yes
No
1.37
-1.35
2.20
x < 6.91
Yes
No
0.9
2.40
x < 3.16
Yes
No
x < 1.36
x < 5.96
Yes
No
Yes
No
x < 0.76
-1.35
2.20
x < 6.91
Yes
No
Yes
No
1.15
1.80
x < 6.31
2.40
Yes
No
1.20
0.60

هرس کردن درخت

- «هرس کردن درخت»: حذف زیردرخت‌ها برای افزایش قابلیت تعمیم‌پذیری
  - Prepruning: در صورتی که تعداد داده‌هایی که به یک گره می‌رسند از یک حد آستانه کم‌تر باشد(به عنوان مثال پنج‌درصد داده‌های آموزشی)، آن گره به گره‌های بیشتر منشعب نخواهد شد.
  - Postpruning: بعد از رشد کامل درخت، زیر درخت‌هایی که موجب بیش‌برازش(Overfitting) می‌شوند، حذف خواهند شد.
    - نیاز به یک مجموعه هرس (pruning set) جدای از مجموعه‌ی آموزشی
- Prepruning سریع‌تر است، در حالی که postpruning دقیق‌تر است.

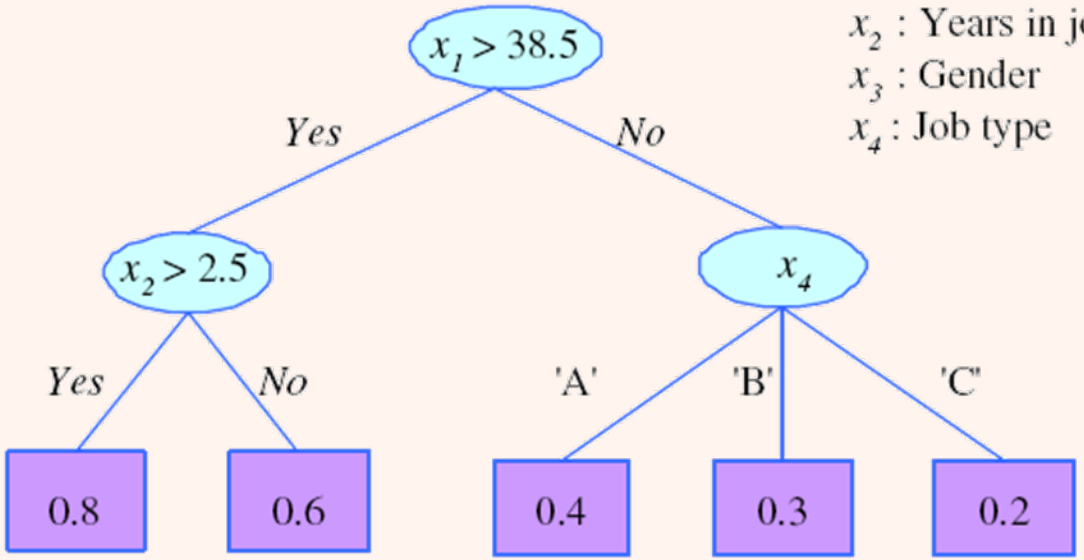


استخراج دانش

- درخت تصمیم به نوعی کار استخراج خصیصه را انجام می‌دهد.
  - ممکن است پس از تشکیل درخت، برخی خصیصه‌ها مورد استفاده قرار نگیرند.
  - می‌توان گفت خصیصه‌هایی که به ریشه نزدیک‌تر هستند، مهمتر می‌باشند.
- درخت تصمیم، قابلیت تفسیرپذیری دارد.
  - مسیر ریشه تا هر برگ، «ترکیب عطفی(conjunction)» یک سری شروط است، چرا که همه‌ی شروط باید برقرار باشند تا داده به گره‌ی برگ مورد نظر برسد.
  - همه‌ی این مسیرها یک «پایگاه قوانین(rule base)» را تشکیل می‌دهند.
  - برای هر قانون می‌توان میزان پشتیبانی(نسبت داده‌های پوشش داده شده توسط قانون) را محاسبه نمود.
  - در دسته‌بندی ممکن است، چند برگ تشکیل یک کلاس را بدهند، در این صورت قوانین به صورت «ترکیب فصلی(disjunction)» بیان می‌شوند.
- می‌توان قوانین به دست آمده را نیز هرس کرد؛ پس از هرس کردن دیگر نمی‌توان درخت معادل آن را ساخت.

استخراج دانش (ادامه...)

x_1 : Age
x_2 : Years in job
x_3 : Gender
x_4 : Job type



R1: IF (age>38.5) AND (years-in-job>2.5) THEN $y = 0.8$
R2: IF (age>38.5) AND (years-in-job≤2.5) THEN $y = 0.6$
R3: IF (age≤38.5) AND (job-type='A') THEN $y = 0.4$
R4: IF (age≤38.5) AND (job-type='B') THEN $y = 0.3$
R5: IF (age≤38.5) AND (job-type='C') THEN $y = 0.2$

به عنوان مثال در صورتی که تمام افرادی که شغل A را دارند، خروجی 0.4 داشته باشد می‌توان R3 را هرس کرد:

$$R3': \text{IF}(\text{job-type='A'}) \text{ THEN } y = 0.4$$

استنتاج قانون

- به جز درخت تصمیم می‌توان قوانین(IF-THEN) را به صورت مستقیم از داده‌ها نیز استخراج نمود.
  - این شیوه یک روش DFS است بر خلاف درخت تصمیم که BFS است:
    - قوانین به ترتیب و یک به یک آموخته می‌شوند.
    - قوانین ترکیب عطفی یک سری شرط هستند.
      - شرط‌ها نیز یک به یک به قوانین افزودن می‌شوند. شرط‌ها به گونه‌ای انتخاب می‌شوند که باعث بهینه‌شدن یک معیار (به عنوان مثال آنتروپی) شوند.
- یک قانون یک نمونه را «پوشش(cover)» می‌دهد، در صورتی که آن نمونه تمام شرایط را ارضا کند.

Sequential covering

- هر قانونی که به اندازه‌ی کافی رشد کرد، پس از هرس شدن به «پایگاه قوانین» اضافه می‌شود، سپس داده‌های پوشش داده شده توسط آن از مجموعه‌ی آموزشی حذف شده و کار با باقی داده‌ها از سر گرفته می‌شود. این کار تا زمانی که همه‌ی داده‌ها پوشش داده شوند ادامه می‌یابد.
  - این الگوریتم به صورت دو حلقه‌ی تودرتوست، حلقه‌ی بیرونی برای افزودن قانون و حلقه‌ی درونی برای افزودن شرط
  - هر دو گام حریصانه هستند و پاسخ بهینه تضمین شده نیست.
  - بعد از هر گام نیز یک مرحله‌ی هرس کردن پیش‌بینی شده است.
- برای حالت دو دسته‌ای(دسته‌ی مثبت و منفی)، قوانین به تدریج افزوده می‌شوند تا دسته‌ی مثبت‌ها پوشش داده شوند؛ در صورتی که داده‌ای با قوانین استنتاج شده پوشش داده نشود، در گروه منفی‌ها قرار می‌گیرد.

Ripper (Cohen, 1995)

IREP (Fürnkrantz and Widmer, 1994)

افزودن شرط

- برای افزودن شرط از معیاری به نام «بهره‌ی اطلاعات (information gain)» استفاده می‌شود:
- پس از افزودن شرط، قانون R به R’ بدل می‌شود:
  - در صورت افزایش بهره شرط به قانون افزوده می‌شود.

به صورت مشابه برای قانون R’

$$Gain(R', R) = s.\left(\log_2 \frac{N'_+}{N'} - \log_2 \frac{N_+}{N}\right)$$

تعداد نمونه‌های مثبت صحیح که با قانون R پوشش داده می‌شود

تعداد کل نمونه‌هایی که با قانون R پوشش داده می‌شود

تعداد نمونه‌های مثبت صحیح در R که با R’ هم مثبت صحیح هستند.

هرس کردن شروط

- تا جایی که هیچ نمونه‌ی منفی پوشش داده نشود به قانون شرط اضافه می‌شود.
- پس از آن نوبت به هرس کردن شروط می‌رسد.
- مبنا افزایش «معیار ارزش قانون» است.

rule value metric

تعداد نمونه‌های مثبت صحیح
(true positive)

تعداد نمونه‌های مثبت کاذب
(false positive)

$$rvm(R) = \frac{p - n}{p + n}$$

- برای این مرحله از «مجوعه‌ی هرس» استفاده می‌شود.

افزودن قوانین

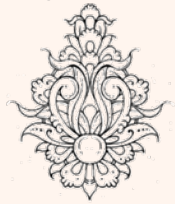
- پس از این یک قانون هرس شد، تمام نمونه‌هایی که توسط آن پوشش داده می‌شود، از مجموعه‌ی آموزشی حذف می‌شود.
- در صورت تهی نبودن مجموعه‌ی آموزشی قانونی دیگر افزوده می‌شود.
- در صورت وجود نویز ممکن است افزودن قانون جدید زودتر متوقف شود.
- ارزش قانون با معیاری «طول توصیف» سنجیده می‌شود.

Description length

- در پایان روی مجموعه قوانین نیز بهینه‌سازی صورت می‌گیرد.

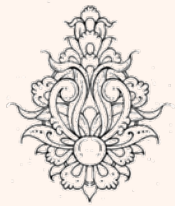
Ripper Algorithm

```
Ripper(Pos,Neg,k)
   RuleSet ← LearnRuleSet(Pos,Neg)
   For k times
      RuleSet ← OptimizeRuleSet(RuleSet,Pos,Neg)
LearnRuleSet(Pos,Neg)
   RuleSet ← ∅
   DL ← DescLen(RuleSet,Pos,Neg)
   Repeat
      Rule ← LearnRule(Pos,Neg)
      Add Rule to RuleSet
      DL' ← DescLen(RuleSet,Pos,Neg)
      If DL'>DL+64
         PruneRuleSet(RuleSet,Pos,Neg)
         Return RuleSet
      If DL'<DL DL ← DL'
         Delete instances covered from Pos and Neg
   Until Pos = ∅
   Return RuleSet
```



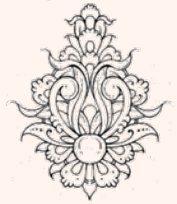
Ripper Algorithm

```
PruneRuleSet(RuleSet,Pos,Neg)
  For each Rule ∈ RuleSet in reverse order
    DL ← DescLen(RuleSet,Pos,Neg)
    DL' ← DescLen(RuleSet-Rule,Pos,Neg)
    IF DL'<DL Delete Rule from RuleSet
  Return RuleSet
OptimizeRuleSet(RuleSet,Pos,Neg)
  For each Rule ∈ RuleSet
      DL0 ← DescLen(RuleSet,Pos,Neg)
      DL1 ← DescLen(RuleSet-Rule+
       ReplaceRule(RuleSet,Pos,Neg),Pos,Neg)
      DL2 ← DescLen(RuleSet-Rule+
       ReviseRule(RuleSet,Rule,Pos,Neg),Pos,Neg)
      If DL1=min(DL0,DL1,DL2)
        Delete Rule from RuleSet and
            add ReplaceRule(RuleSet,Pos,Neg)
      Else If DL2=min(DL0,DL1,DL2)
        Delete Rule from RuleSet and
            add ReviseRule(RuleSet,Rule,Pos,Neg)
  Return RuleSet
```



حالت بیش از دو دسته

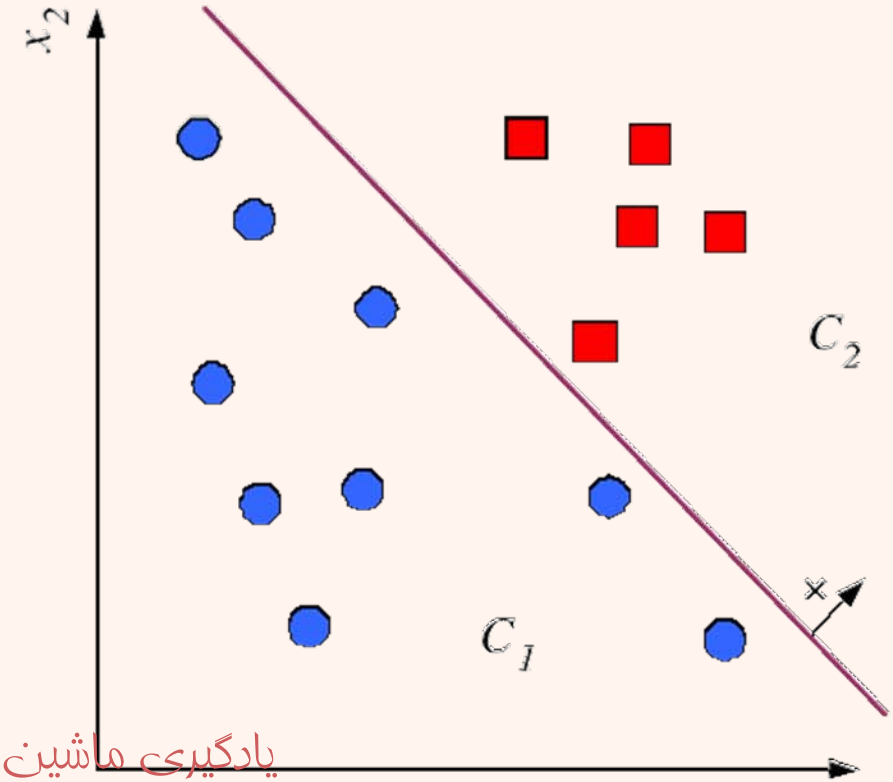
- در این حالت کلاس‌ها به ترتیب احتمال رخداد آن‌ها مرتب می‌شوند، به گونه‌ای که C_1 کم‌ترین احتمال رخداد و C_k بیشترین احتمال را داشته باشند.
- یک مساله‌ی دو کلاسه (C_1 و سایر کلاس‌ها) تعریف و به شیوه‌ی گفته شده بررسی می‌شود. پس از استخراج قانون برای کلاس C_1 و حذف نمونه‌های مرتبط با آن همین روند برای سایر کلاس‌ها ادامه می‌یابد.

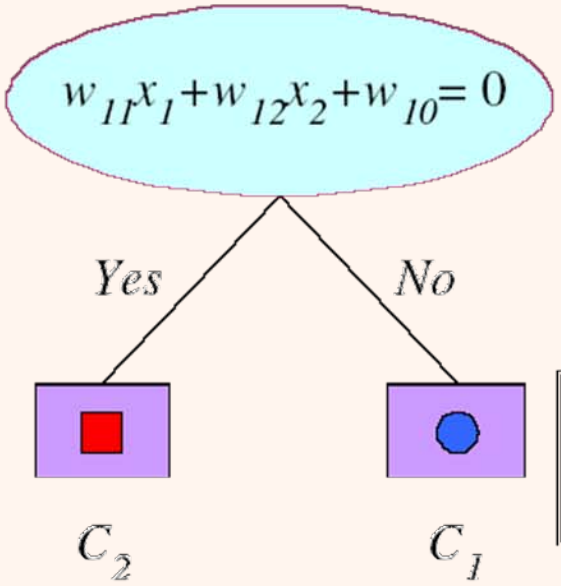


درخت چندمتغیره

- در درخت چندمتغیره تمام خصیصه‌های در گره‌های داخلی قابل استفاده هستند، به عنوان مثال برای خصیصه‌های پیوسته:

$$f_m(\mathbf{x}) = \mathbf{w}_m^T \mathbf{x} + w_{m0} > 0$$





درخت چندمتغیره(ادامه...)

- بدین‌ترتیب، نواحی محلی با یک چندضلعی محدود می‌شوند.(در حالت تک‌متغیره، جداساز i، بر محور i عمود است)
  - در حالت تک‌متغیره، تنها d راستا برای انتخاب وجود دارد و N_m-1 مقدار برای حدآستانه
- در حالت چندمتغیره، امکان استفاده از گره‌های غیرخطی وجود دارد.
- همچنین استفاده از MLP در هر گره، پیشنهاد شده است.

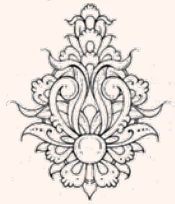
$$f_m(\mathbf{x}) = \mathbf{x}\mathbf{W}_m^T\mathbf{x} + \mathbf{w}_m^T\mathbf{x} + w_{m0} > 0$$

- همچنین استفاده از گره‌های کروی (sphere node) پیشنهاد شده است.

$$f_m(\mathbf{x}) = \left\|\mathbf{x} - \mathbf{c}_m\right\| > \alpha_m$$

درخت چندمتغیره(ادامه...)

- برای CART یک نمونه‌ی چندمتغیره پیشنهاد شده است.
- در برخی روش‌ها، مرحله‌ی پیش‌پردازش کاهش بعد نیز انجام می‌شود.
- در روش تک‌متغیره، فضای فرضیه ساده‌تر است، در نتیجه در فضای چندمتغیره با تعداد گره‌های کم‌تر تخمین بهتری انجام می‌شود.
  - تعداد انشعاب‌ها(branching factor) هم اثر مشابهی دارد.
- بین ابعاد درخت، تعداد انشعاب‌ها و پیچیدگی گره‌ها trade-off وجود دارد.



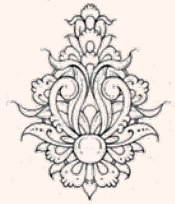
درخت چندمتغیره(ادامه...)

یک درخت بزرگ با گره‌های ساده و تعداد انشعاب کم

یا

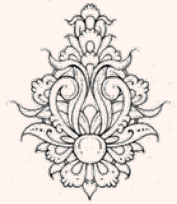
یک درخت کوچک با گره‌های پیچیده و تعداد انشعاب زیاد

- قابلیت تفسیرپذیری و تعمیم‌پذیری درخت اول بهتر است.
- یکی از مزایای درخت، شکستن مسأله به مسائل کوچک‌تر است، افزایش پیچیدگی در واقع نقض غرض است.



Omnivariate Decision Trees

- دارای ساختار ترکیبی است.
  - در هر گره، با توجه به نوع و پیچیدگی مسأله، جداساز مطلوب ممکن است متفاوت باشد.
  - گره‌ها ممکن است تک‌متغیره، چندمتغیره(خطی و یا غیرخطی) باشد.
  - مدل‌های پیچیده تنها زمانی انتخاب می‌شوند که کارایی مدل‌های ساده کاریی قابل مقایسه نداشته باشند.
  - نتایج نشان داده است که گره‌های نزدیک به ریشه پیچیده‌تر هستند(هرچه از ریشه دور شویم، مسأله ساده‌تر می‌شود).



مزایای درخت تصمیم

- مشابه سایر روش‌های ناپارامتری، برگ‌ها معادل نوارها هستند؛ با این تفاوت که نه عرض نوارها ثابت است و نه تعداد نمونه‌ها
- تقسیم‌بندی نواحی محلی، تنها بر اساس شباهت نمونه‌ها انجام نمی‌شود، بلکه خروجی هم در نظر گرفته می‌شود.
- سرعت بیشتری دارند.
- نیاز به ذخیره‌ی همه‌ی نمونه‌ها نیست.