

観察からの学習

Chapter 18

Section 1 – 3,5

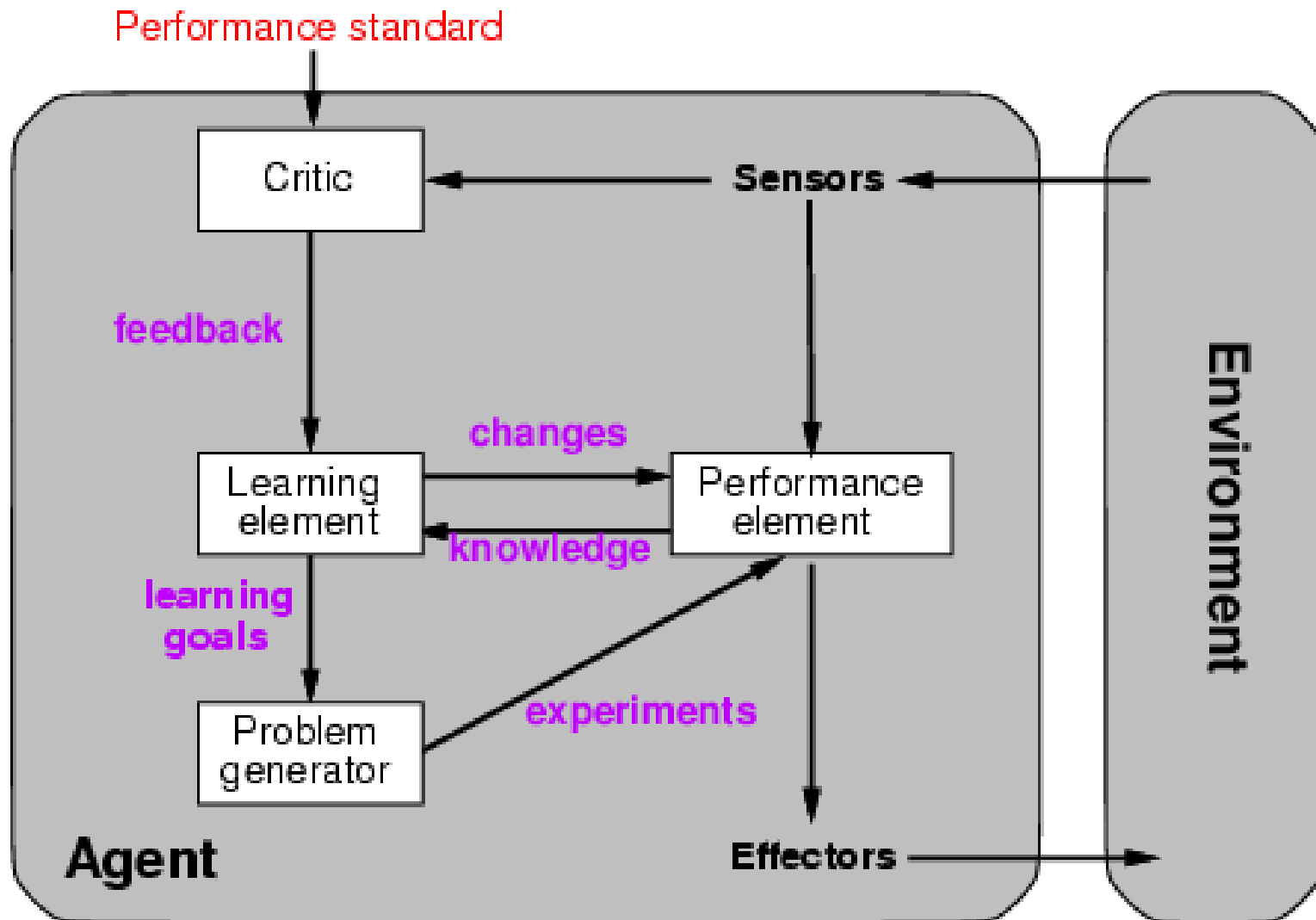
概要

- 学習エージェント
- 帰納的学習
- 決定木学習

学習

- 学習は未知の環境では本質的
 - 設計者が全能でないときと同値
- 学習はシステム構成の方法として有用
 - その方法を書き下そうとするよりもエージェントを現実に立ち向かわせる
- 学習は性能を向上させるようにエージェントの決定機構を修正させる

Learning agents

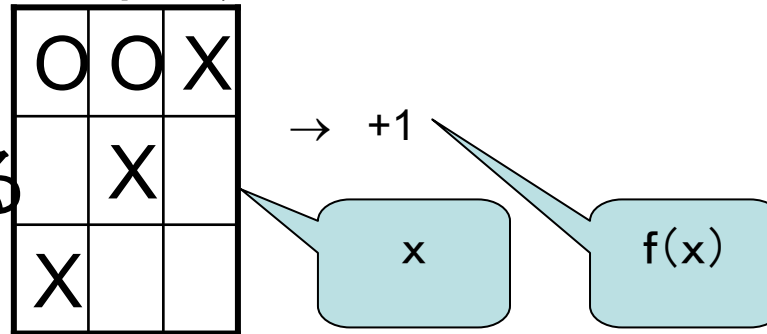


学習要素

- 学習要素の設計は次のものに影響される
 - 性能要素のどの部分を学ぶか
 - この部分を学ぶためにどのようなフィードバックが用意されているか
 - この部品を使うためにどのような表現が使われるか
- フィードバックの種類:
 - 教師あり学習：それぞれの事例に対して正解が用意
 - 教師なし学習：正解が与えられない
 - 強化学習：時々報酬

帰納的学習

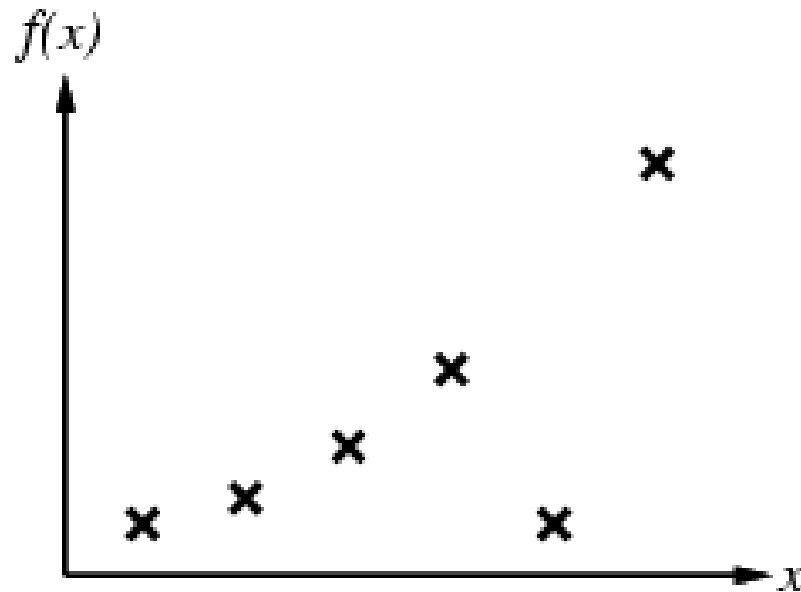
- 最も簡単な方法: 事例から関数を学ぶ
- f は目標関数
- 事例は対 $(x, f(x))$ である



- 問題: 事例による訓練用の集合を与えられたとき、 $h \approx f$ とするような仮説 h を発見する
- (これは実際の学習を極端に簡易化したもの):
 - 前もっての知識を無視している
 - 事例が与えられると仮定している

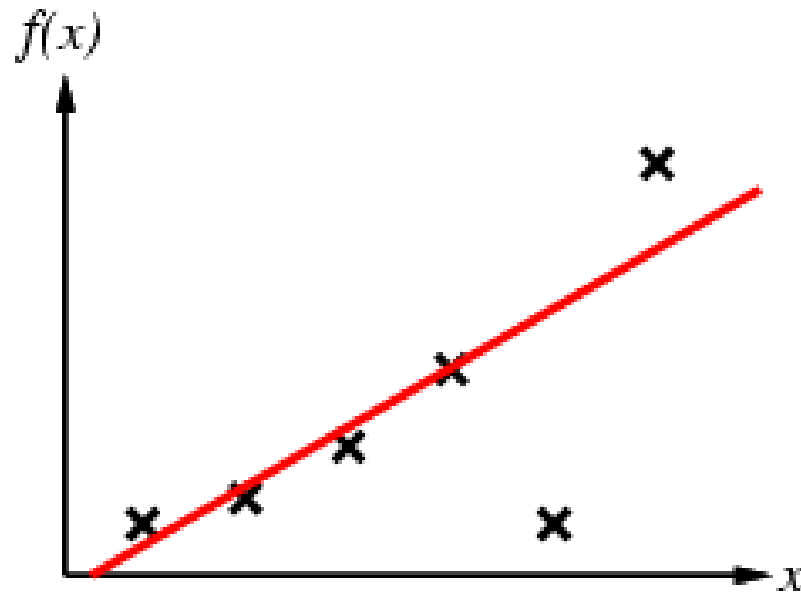
帰納的学習の方法

- 訓練集合で f に一致するように h を構成・調整する
- (事例で h が f に一致するなら、 h は一貫性がある)
- 例 カーブをあわせる:



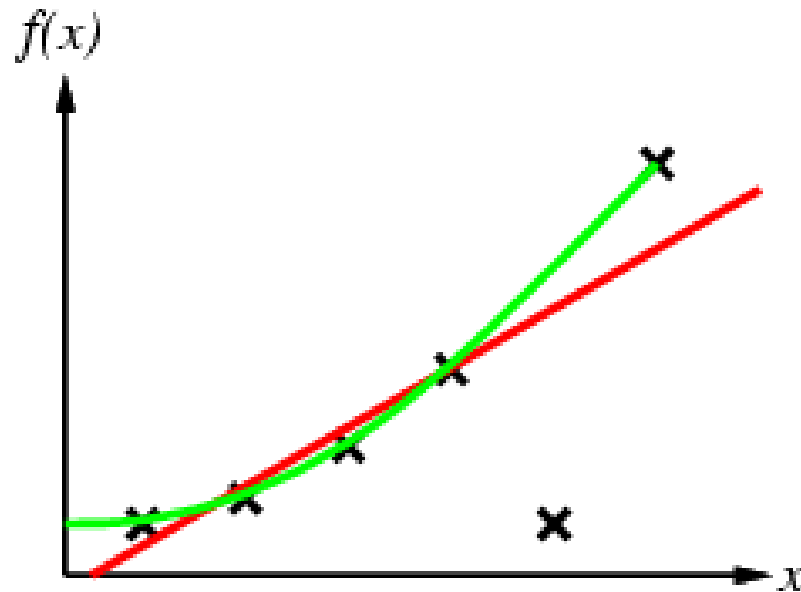
帰納的学習の方法

- 訓練集合で f に一致するように h を構成・調整する
- (事例で h が f に一致するなら、 h は一貫性がある)
- 例 カーブをあわせる:



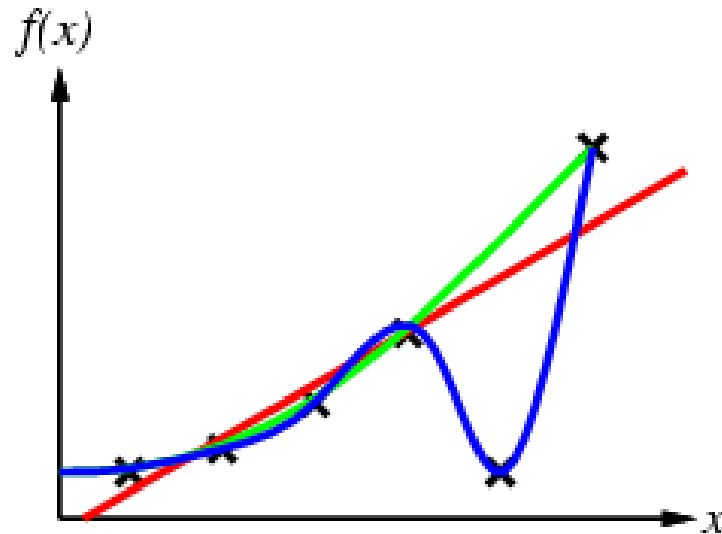
帰納的学習の方法

- 訓練集合で f に一致するように h を構成・調整する
- (事例で h が f に一致するなら、 h は一貫性がある)
- 例 カーブをあわせる:



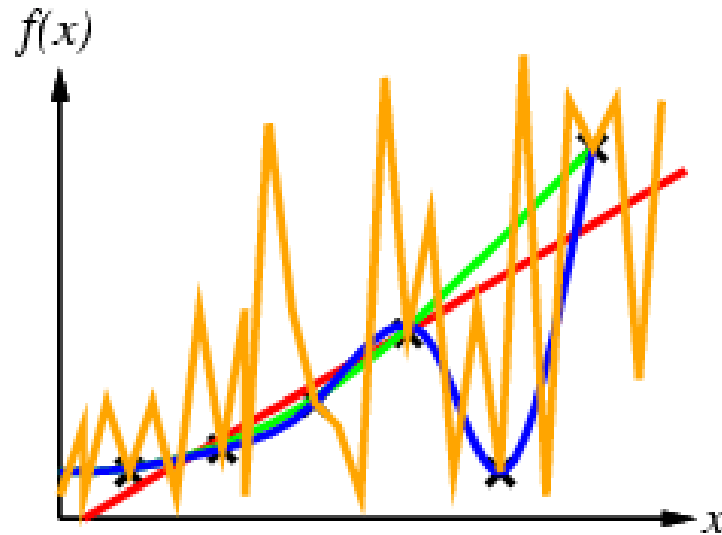
帰納的学習の方法

- 訓練集合で f に一致するように h を構成・調整する
- (事例で h が f に一致するなら、 h は一貫性がある)
- 例 カーブをあわせる:



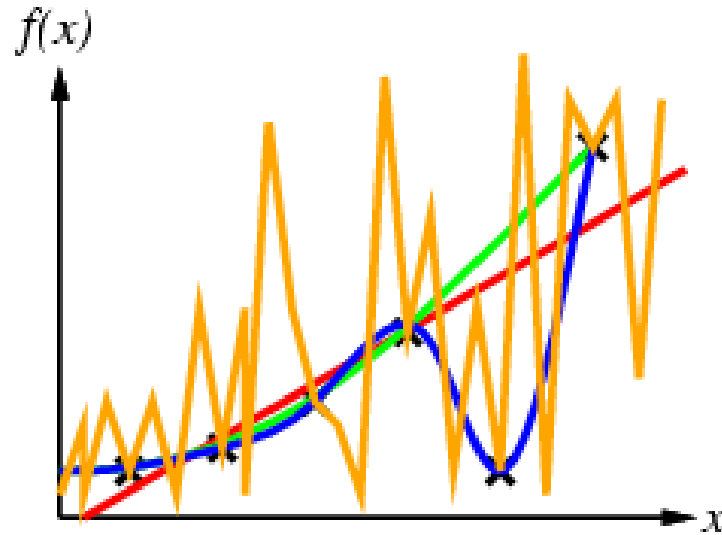
帰納的学習の方法

- 訓練集合で f に一致するように h を構成・調整する
- (事例で h が f に一致するなら、 h は一貫性がある)
- 例 カーブをあわせる:



帰納的学習の方法

- 訓練集合で f に一致するように h を構成・調整する
- (事例で h が f に一致するなら、 h は一貫性がある)
- 例 カーブをあわせる:



- オッカムのかみそり：データと一貫性の有る最も簡単な仮説を好む

学習決定木

- 問題: 次の属性の元で、レストランで席を待つべきかを決める:
 1. 代替: 近くに代替りのレストランがあるか
 2. バー: 待つ間、居心地のよいバーの場所があるか
 3. 金土: 今日は金曜かあるいは土曜か
 4. 空腹: われわれは空腹か
 5. 客: レストランには何人の客がいるか (空, 有る程度, 満席)
 6. 価格: 価格の範囲(\$, \$\$, \$\$\$)
 7. 雨: 外は雨か
 8. 予約: 予約をしたか
 9. 種類: 何料理か (フランス、イタリア、タイ、バーガー)
 10. 待ち時間: どのくらい待たされるか(0-10, 10-30, 30-60, >60)

属性ベースの表現

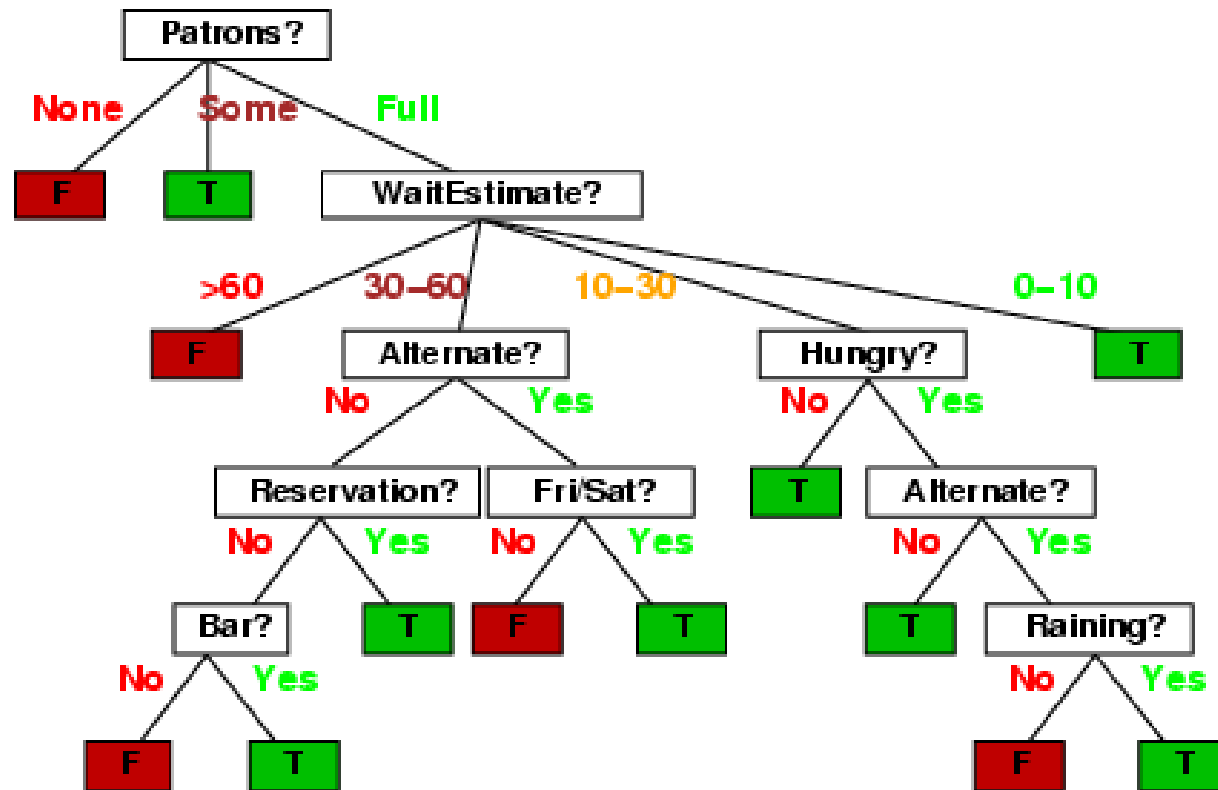
- 事例が属性の値(Boolean, discrete, continuous)によって示される
- 席を待つ、待たないの状況:

Example	Attributes										Target
	<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>Wait</i>
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

- 事例でのクラス分けは肯定(T)か否定(F)

決定木

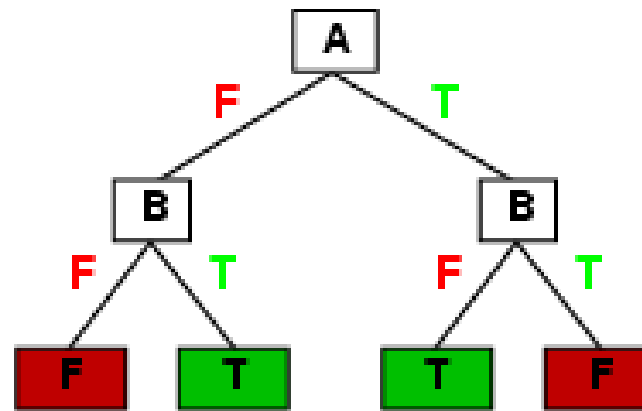
- 仮設の一つの表現法
- 以下は待つかどうかについての肯定の木



表現力

- 決定木は入力属性のいかなる関数をも表すことが可能
- ブール関数に対しては真理表の行 → 葉へのパス:

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



- いかなる訓練集合に対しても一貫性のある決定木があり、この木は各事例 (x で f が非決定的でない限り) に対して葉へのパスを有する。しかし、新しい事例を作り出すことはない
- もっと簡便な決定木を探したい

仮説空間

- n のブール属性に対してどれだけ異なる決定木が存在する
のか
 - = ブール関数の数
 - = 2^n 行 = 2^{2^n} の異なる真理表
- 6個のブール属性で18,446,744,073,709,551,616木

仮説空間

- n のブール属性に対してどれだけ異なる決定木が存在するのか
 - = ブール関数の数
 - = 2^n 行 = 2^{2^n} の異なる真理表
- 6個のブール属性で18,446,744,073,709,551,616木
- 単なる連言の仮説はいくつになるか (e.g., $Hungry \wedge \neg Rain$)
- 属性は 内部(肯定), 内部(否定)あるいは外部の値を取る
 - ⇒ 3^n 個の異なる連言の仮説
- より表現的な仮説の空間
 - 目的関数が表される機会を増やす
 - 訓練集合と一貫性の有る仮説の数を増やす
 - ⇒ より悪い予測をもたらすことも

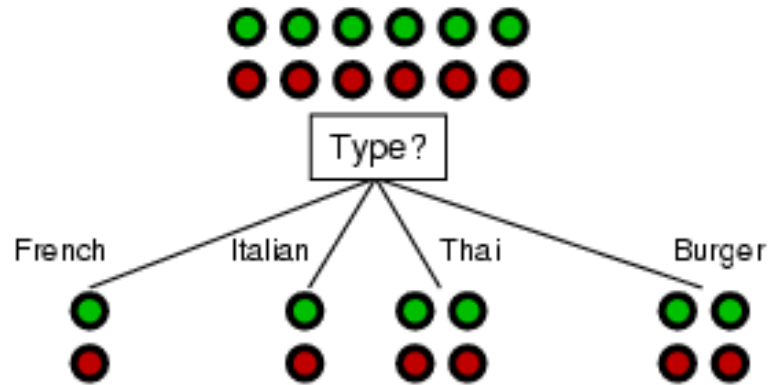
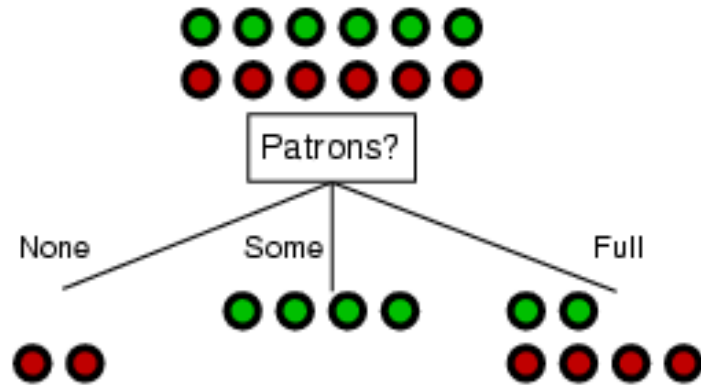
決定木学習

- 目的: 訓練用のサンプルと一貫性のある小さな木を見つける
- 考え方: (再帰的に) 木のルートとして“最も重要な”属性を選ぶこと

```
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
      examplesi ← {elements of examples with best =  $v_i$ }
      subtree ← DTL(examplesi, attributes – best, MODE(examples))
      add a branch to tree with label  $v_i$  and subtree subtree
  return tree
```

属性の選択

- 考え方: よい属性は事例を理想的に“全て肯定”と“全て否定”に分割する



- *Patrons?* はよい選択か

情報定理を用いて

- DTLアルゴリズムでChoose-Attribute を実現するために
- 情報の内容 (エントロピー):

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

- p 個の肯定の事例と n 個の否定の事例を含んでいる訓練集合に対して:

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

情報定理を用いて

- 選択した属性Aは訓練の集合Eを、Aに対する値に従って部分集合 E_1, \dots, E_v に分割する。ここではAはv個の異なる値を取る

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

- 属性テストでのエントロピーによる情報の獲得(IG)あるいは減少は:

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

- IGを最大にする属性を選択する

情報の獲得

- 訓練集合に対して, $p = n = 6$, $I(6/12, 6/12) = 1$ bit
- 属性 *Patrons* と *Type* を考える (そして他も):

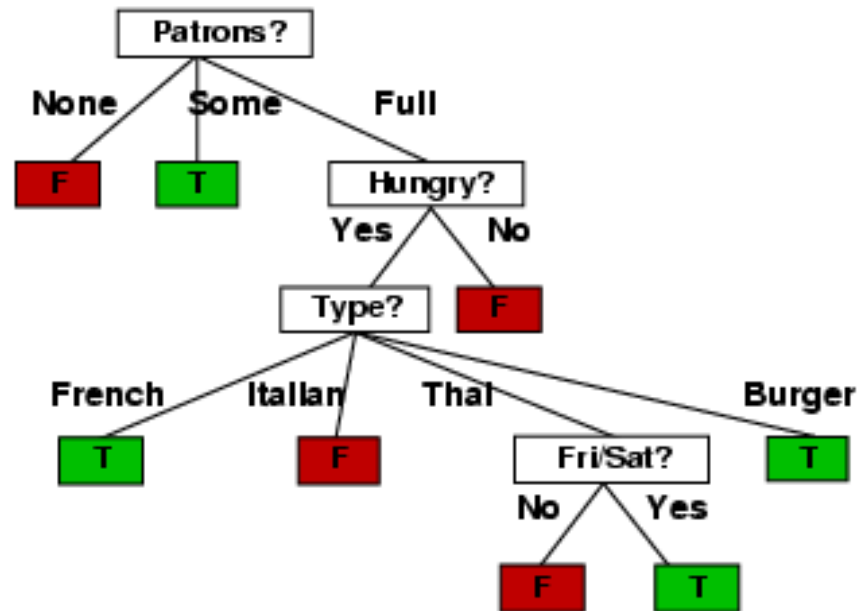
$$IG(\textit{Patrons}) = 1 - \left[\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I\left(\frac{2}{6}, \frac{4}{6}\right) \right] = .0541 \text{ bits}$$

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

- *Patrons* が全ての属性の中で最大のIGであるので DTL アルゴリズムは *Patrons* をルートにする

Example

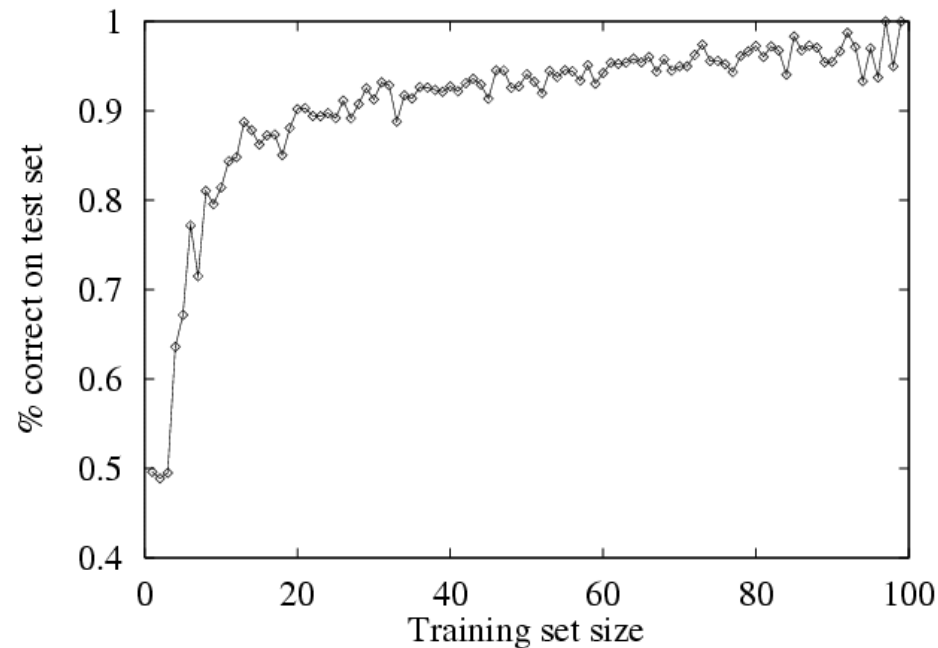
- 12個の事例から学習した決定木:



- “肯定”の木より簡単。より複雑な仮設は少量のデータによって正当化されない

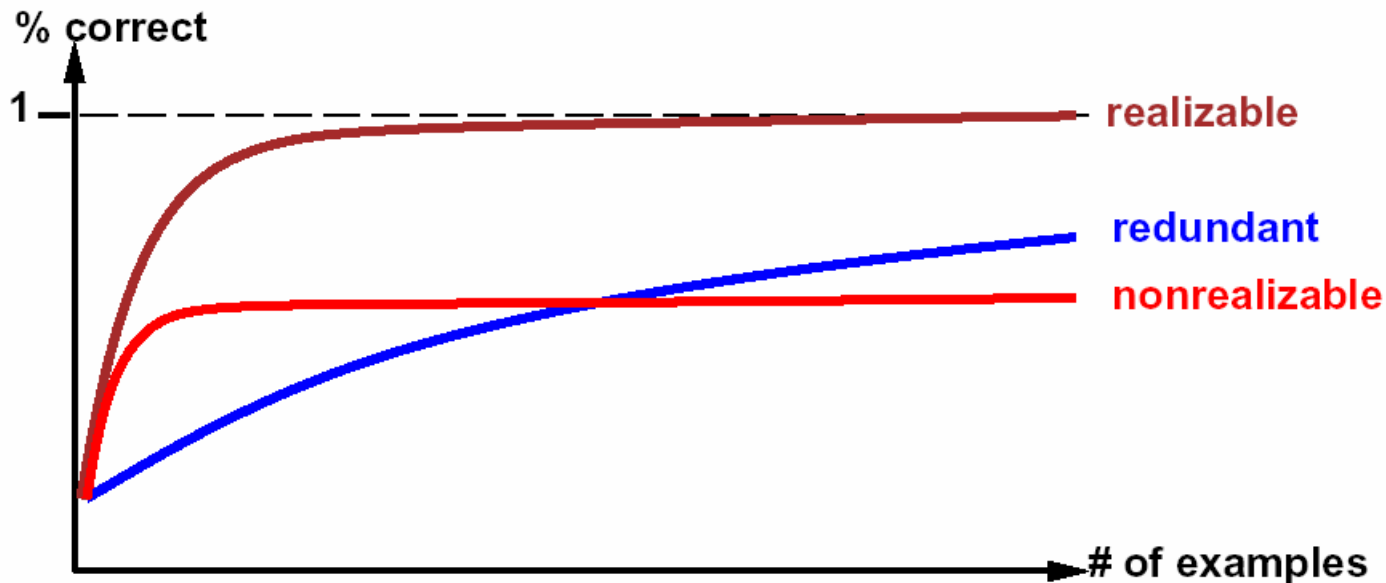
性能測定

- $h \approx f$ であることをどのように知るか
 - 計算可能・統計的学習の定理を用いる
 - 事例による新たな**テスト集合**で h を試みる
- (訓練集合で用いた事例の空間と**同一**の分布を用いる)
- **学習カーブ** = 訓練集合の大きさの関数としてのテスト集合の正しさの割合



学習性能について

- 学習カーブは次のことに従属する
 - 実現可能 (目的関数を表すことができる) か実現不可能
 - 実現不可能は属性の欠如による
 - あるいは制約された仮説のクラス (例えば頭打ちの線形関数)
 - 冗長な表現(例えば無関係な属性からの負荷)



計算論的学習定理

- 特定の訓練集合から組み立てた仮説が実際の関数に十分に近いと知ることができるのか
- いくつの事例が十分な量か。どのように選ぶべきか
- 訓練集合の大きさと確率的分布に関連して学習アルゴリズムはどれだけよいのかを示してくれる計算論的性格を知りたい

PAC learning

- おそらく近似的に正しい仮定 (PAC): 非常に悪い仮説 h は非常に高い確率で早い段階で修正される [Valiant 1984].
- 不動集合の仮定: 訓練集合とテスト集合は同じ確率分布を用いて同じ事例の集まりからランダムに抽出される
- 集合の大きさ、実際にそして期待される誤り、仮説空間の大きさには関係が有る

PAC学習：形式化

- \mathbf{X} は可能な全ての事例の集合
- D は事例が抽出されたところの分布
- \mathbf{H} は可能な全ての仮説空間で $f \in \mathbf{H}$
- m は訓練のための事例の数
- $error(h) = \text{Probability}(h(x) \neq f(x) \mid$
 $x \text{ is drawn from } \mathbf{X} \text{ with } D)$
- $error(h) \leq \varepsilon$ ならば h は近似的に正しい

PAC学習：形式化

- 証明すべきこと: m 個の事例の後、高い確率で全ての一貫性のある仮説は近似的に正しい
- 全ての一貫性のある仮説は f の近くの半径 ε の中にある



複雑性の解析

- 非常に悪い仮説 $h_{\text{bad}} \in \mathbf{H}_{\text{bad}}$ が最初の m 個の事例と一貫性がある確率は
 - 定義により $\text{error}(h_{\text{bad}}) > \varepsilon$
 - 一つの事例と一致する確率は $(1 - \varepsilon)$ であり、 m 個の事例とは $(1 - \varepsilon)^m$
- \mathbf{H}_{bad} が一貫性がある仮説を少なくとも一つ有している確率は

Probability(\mathbf{H}_{bad} has a consistent hypothesis)

$$\leq |\mathbf{H}_{\text{bad}}|(1 - \varepsilon)^m \leq |\mathbf{H}|(1 - \varepsilon)^m$$

非常に悪いがすべての事例と合致するようなことがあってはならない

非常に悪い仮設の数

仮設の数

あまりよい類推とは考えにくいがほかに方法がないのであろう

複雑性の解析

- この値をある小さな確率 δ に抑えたいとする

$$|H|(1-\varepsilon)^m \leq \delta$$

- これは少なくとも m 個の事例が次のようになっているとき可能である

$$m \geq 1/\varepsilon(\ln 1/\delta + \ln |H|)$$

これは事例の複雑性である

- “このたくさんの事例 m で一貫性がある仮説を学習アルゴリズムが返すなら、少なくとも確率 $1-\delta$ でエラーは高々 ε である”
- $|H| = 2^{2^n}$ なので複雑性は属性 n の数に応じて指数的(2^n)に大きくなる
- 結論: ブール関数を学習することはテーブルルックアップよりは良くはない

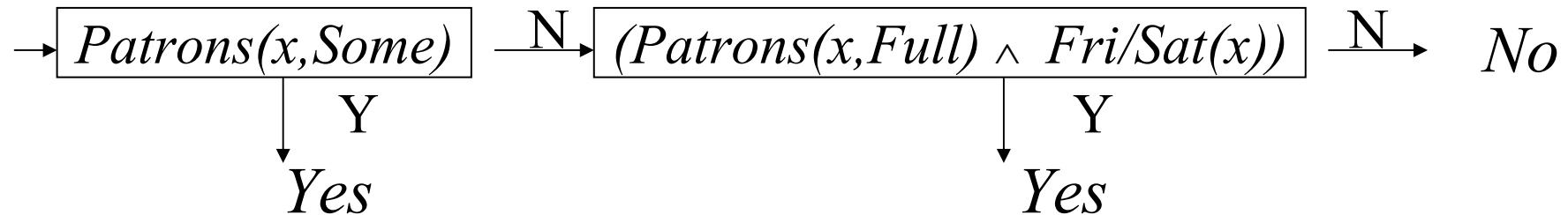
PAC学習：観察

- “仮説 $h(X)$ は m 個の事例と一貫性が有り確率 $1 - \delta$ で最大 ε のあやまりがある
- これは最悪の場合の解析。結果は分布 D には独立である
- 拡大率の解析:
 - for $\varepsilon \rightarrow 0$, $m \rightarrow \infty$ proportionally
 - for $\delta \rightarrow 0$, $m \rightarrow \infty$ logarithmically
 - for $|\mathbf{H}| \rightarrow \infty$, $m \rightarrow \infty$ logarithmically

決定リストを学習する

- 可能な仮設の空間を制限する
 - 最も簡単な仮説を返す – 解決困難!
 - ブール関数の部分集合のみを考える
- 決定木は制限された形式での論理的表現:

$$\forall x \text{ WillWait}(x) \iff \text{Patrons}(x, \text{Some}) \vee (\text{Patrons}(x, \text{Full}) \wedge \text{Fri/Sat}(x))$$



決定リストを学習する

- 任意の大きさのリストはあるブール関数を表すことができる。たかだか $k < n$ リテラルのテストを伴ったリストは k -DLブール言語を定義する。 n 属性では言語は k -DL(n)
- テストの言語 $Conj(n,k)$ はたかだか $3^{|Conj(n,k)|}$ の異なったコンポーネントの集合を持つ (Y,N,absent)
- $|k\text{-DL}(n)| \leq 3^{|Conj(n,k)|} |Conj(n,k)|!$ (any order)
- $|Conj(n,k)| = \sum_{i=0}^k \binom{2n}{i} = O(n^k)$

決定リストを学習する

- $|k\text{-DL}(n)| = 2^{O(n^k \log(n^k))}$
- m 個の式での置き換えを行うこと:
$$m \geq \frac{1}{\varepsilon} (\ln \frac{1}{\delta} + O(n^k \log(n^k)))$$
- ちいさな数 k に対して合理的であるリテラルテストの大きさでは事例の数は多項式である
- アルゴリズム: 訓練集合にまったく一致するテストを見出し、それを決定リストに加え、事例から除く。全ての事例が取り除かれるまでこれを続ける

決定リスト学習アルゴリズム

function Decision-List-Learning(*examples*)

returns a decision list *DL*, *No*, or *Fail*

if *examples* is empty **then return** *No*

t := a test that matches a nonempty subset of *examples_i* of examples such that all elements in it are either positive or negative

if there is no such *t* **then return** *Fail*

if the examples in *examples_i* are positive **then** *o* := *Yes*

else *o* := *No*

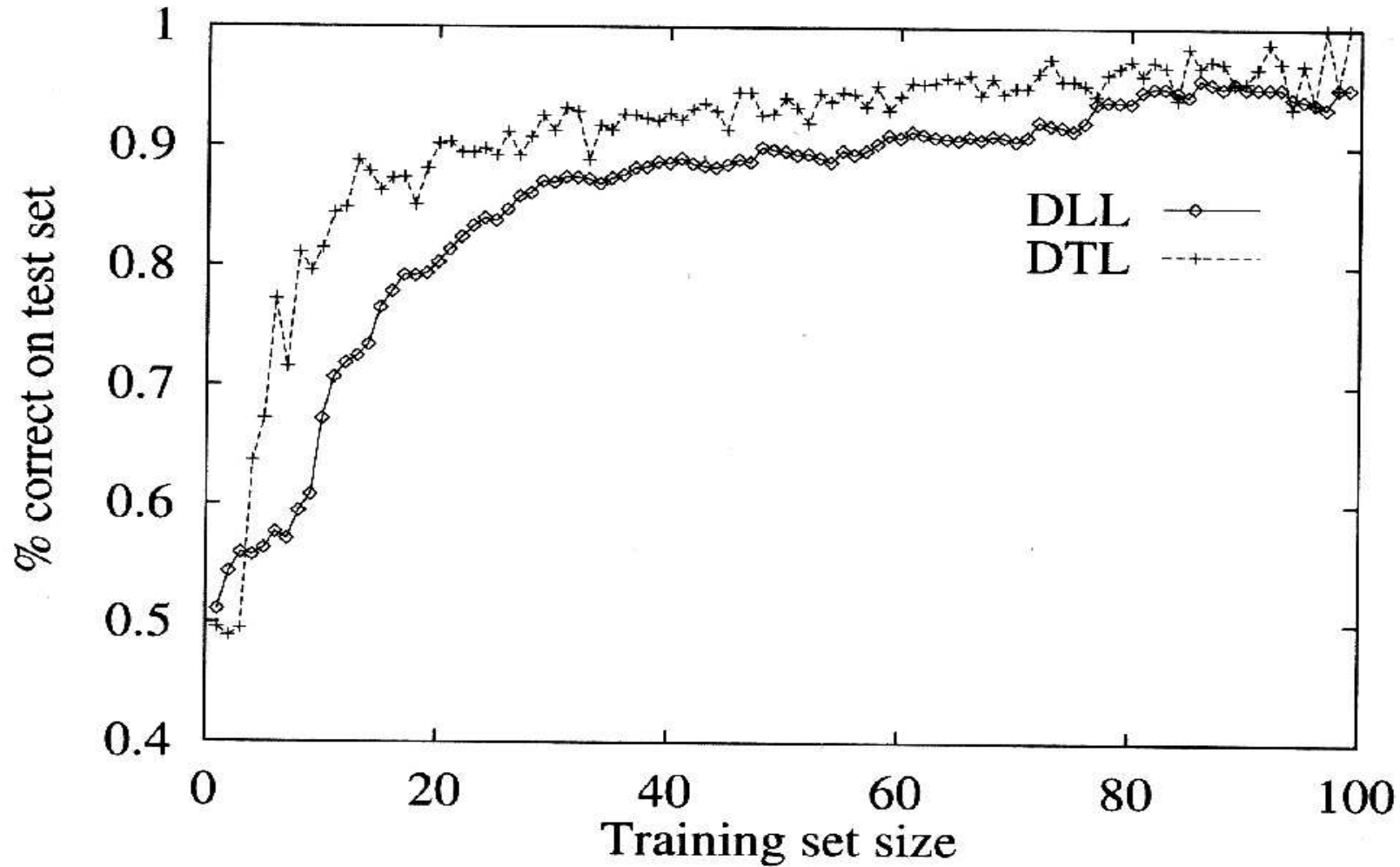
return a decision list *DL* with initial test *t* and outcome *o* and remaining elements defined by

Decision-List-Learning(*examples* - *examples_i*)

席を待つ例

Example	Attributes										Target Wait
	<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>	
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

席を待つ例



制限されたブール関数

Type	Complexity
Boolean conjunctions	<i>polynomial</i>
K-DNF	<i>polynomial</i>
K-CNF	<i>polynomial</i>
K-DL	<i>polynomial</i>
Smallest K-DL	<i>NP-hard</i>

バイアスの種類

- 絶対バイアス
 - 仮説の種類を限定する: CNF, k-DL, etc
- 好みのバイアス
 - when h_1 と h_2 a 化一貫する仮説のとき、簡単な方 (短い方, 最小の木, ...)
- ランダムバイアス
 - 一貫している配列野中からランダムに一つ抽出する
- 悪いバイアスからの復帰
 - “平均の” 多重の配列
 - 異なったバイアスに自然に変化する

まとめ

- 学習は未知の環境と怠惰な設計で必要とされる
- 学習エージェント = 性能要素 + 学習要素
- 教師あり学習では、訓練用の事例と一貫する最も簡単な仮説を見つけること
- 情報獲得を用いての決定木学習
- 学習性能 = テスト集合で計測されて予測の正確さ