CSCM603130: Sistem Cerdas
Machine Learning

Fariz Darari, Aruni Yasmin Azizah

Fakultas Ilmu Komputer
Universitas Indonesia

2019/2020 • Semester Ganjil

Outline

Outline

Learning Agents

Dua pendekatan membangun agent:

- Dirancang, diprogram, diberi knowledge oleh manusia
- Dirancang sehingga bisa belajar dari input (percept, pengalaman, dst.)

Mengapa learning agent dibutuhkan:

Learning Agents

Dua pendekatan membangun agent:

- Dirancang, diprogram, diberi knowledge oleh manusia
- Dirancang sehingga bisa belajar dari input (percept, pengalaman, dst.)

Mengapa learning agent dibutuhkan:

- Environment yang bervariasi dan berubah seiring berjalannya waktu.
- *Agent designer* tidak dapat mengetahui semua solusi.

Agent yang Belajar

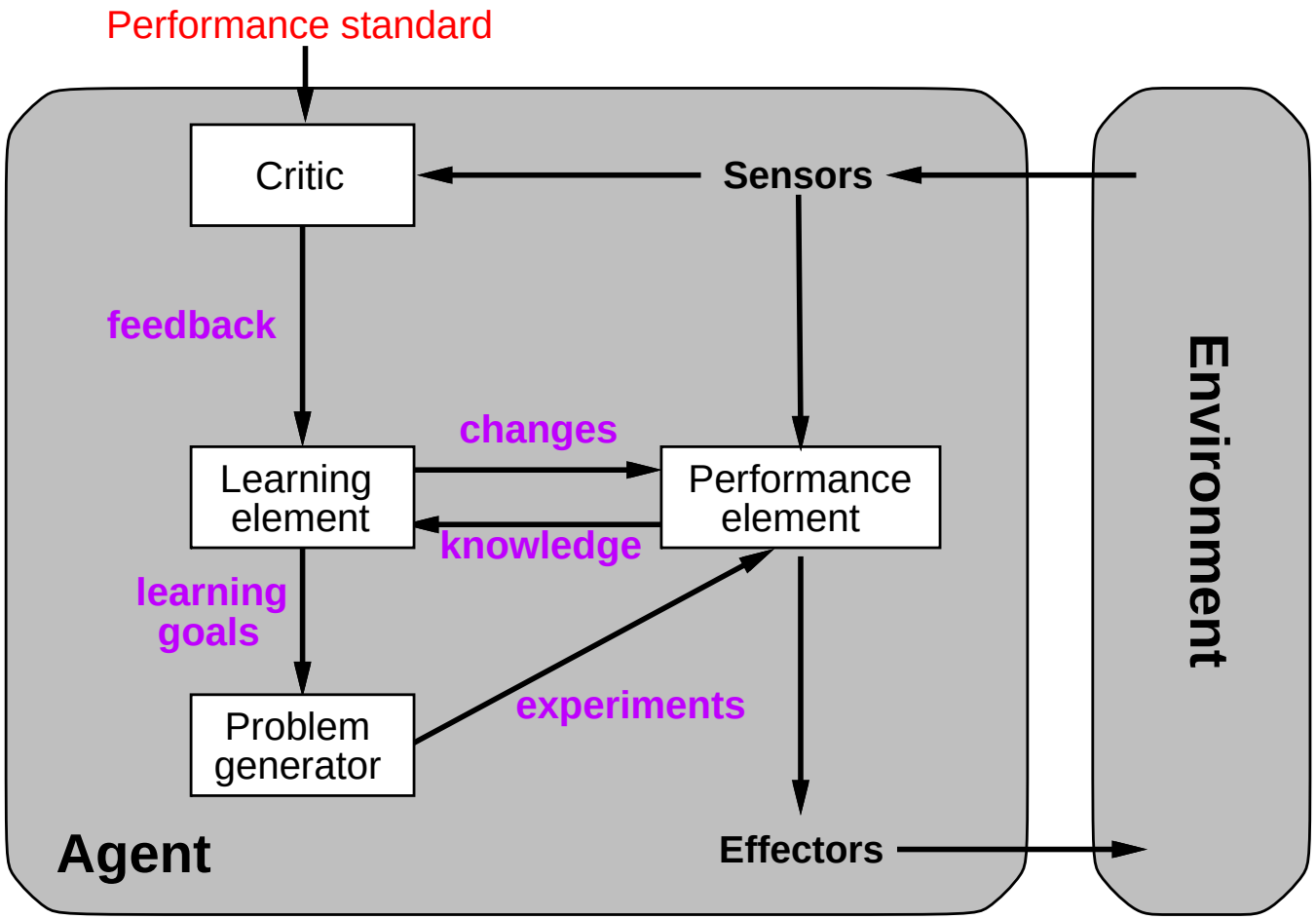
Kita telah mengenal banyak jenis agent:

- Simple reflex agent (condition-action rules)
- Goal-based agent (memiliki goal):
  - Problem-solving agent: searching
  - Knowledge-based agent: knowledge representation & reasoning dengan pendekatan logika dan probabilistik
- Utility-based agent (mengukur nilai utility sebuh state)

Semua agent ini bisa dibangun dengan metode "pemelajaran" yang tepat!

Contoh: agent supir taksi. Bagaimana dia belajar?

Arsitektur Learning Agent



Jenis Metode Learning

Supervised learning

Agent belajar fungsi yang memetakan input ke output

- Pada tahap *training*, learning algorithm menerima pasangan input-output yang spesifik.
- Sample ini dipakai untuk estimasi fungsinya.

Contoh:
Sebuah agent taxi diberikan informasi mengenai lampu sen (input). Jika mobil yang berada di depan agent taxi menyalakan lampu sen, maka taxi harus diperlambat (output).
Contoh task: regression, classification

Jenis Metode Learning

Unsupervised learning

Sebuah learning algorithm menerima sekumpulan data, dan harus menemukan pola-pola di dalamnya.

Contoh:
Sebuah agent taxi menerima data mengenai laju lalin sepanjang hari. Mungkin ia bisa belajar periode "morning rush hour", "evening rush hour"
Contoh task: *Clustering*

Jenis Metode Learning

Reinforcement learning

- Sebuah agent menerima input data dan harus mengambil tindakan.
- Agent lalu menerima reinforcement signal (berupa *reward* atau *punishment* mis. good, bad) sebagai akibat tindakan.
- Learning algorithm memodifikasi agent function untuk memaksimalkan signal "good".
- **Contoh**:
  Ketika taxi agent mendapat nilai tip yang sedikit setelah mengantar penumpang, mengindikasikan taxi agent melakukan kesalahan.

Jenis Metode Learning

Semi-supervised learning

- Learning algorithm menerima beberapa (*labeled*) input yang memiliki pasangan output (baik akurat maupun *noise*).
- Kemudian, learning algorithm harus memberikan output untuk banyak input baru (*unlabeled*).
- Kesatuan dari *supervised learning* dan *unsupervised learning*.
- **Contoh**:
  Agent untuk menebak usia seseorang dari fotonya. Agent menggunakan input beberapa foto orang, beserta usia yang dilaporkan orang tsb, dan usia sebenarnya.

Outline

Supervised Learning

Supervised Learning

Prinsip dasar dari supervised learning adalah mempelajari fungsi atau aturan dari pasangan input-output atau **"belajar dari pengalaman"** (*inductive learning*).

Metode ilmiah empiris bisa dilihat sebagai proses *inductive learning*:

Supervised Learning

Supervised Learning

Prinsip dasar dari supervised learning adalah mempelajari fungsi atau aturan dari pasangan input-output atau **"belajar dari pengalaman"** (*inductive learning*).

Metode ilmiah empiris bisa dilihat sebagai proses *inductive learning*:

1. Lakukan ujicoba, kumpulkan data
2. Rumuskan hipotesis yang konsisten dengan data
3. Hipotesis ini memprediksi nilai data baru
4. Lakukan ujicoba untuk memeriksa kebenaran prediksi
5. Tambahkan ke data yang kita miliki
6. Jika hipotesis tidak lagi konsisten, rumuskan ulang!

Supervised Learning

Ide dasar supervised learning:

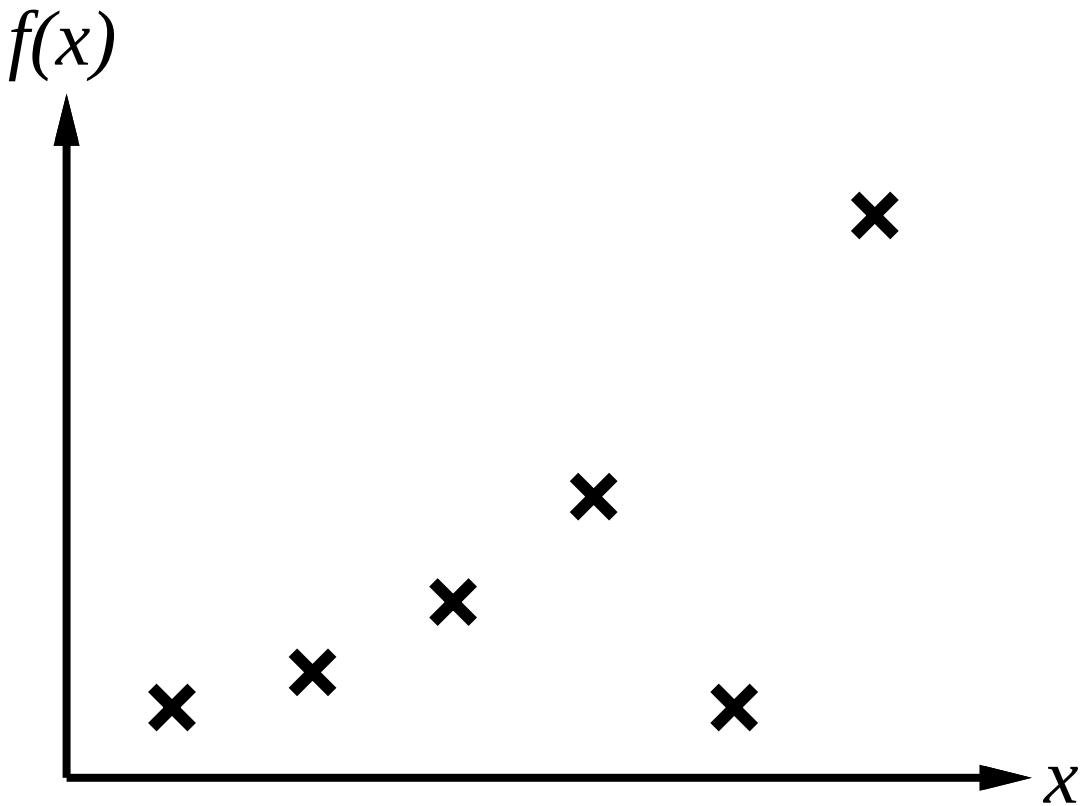
- Diberikan sebuah training set (example) N berupa pasangan input-output: $(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$, dimana y_j dihasilkan dari suatu fungsi yang tidak diketahui $y = f(x)$.
- Cari fungsi hipotesis h yang mengaproksimasi f.

- Fungsi h yang bagus bisa memprediksi *example* yang belum dilihat pada saat belajar.
- Jika domain output y diskrit: classification
- Jika domain output y kontinu: regression

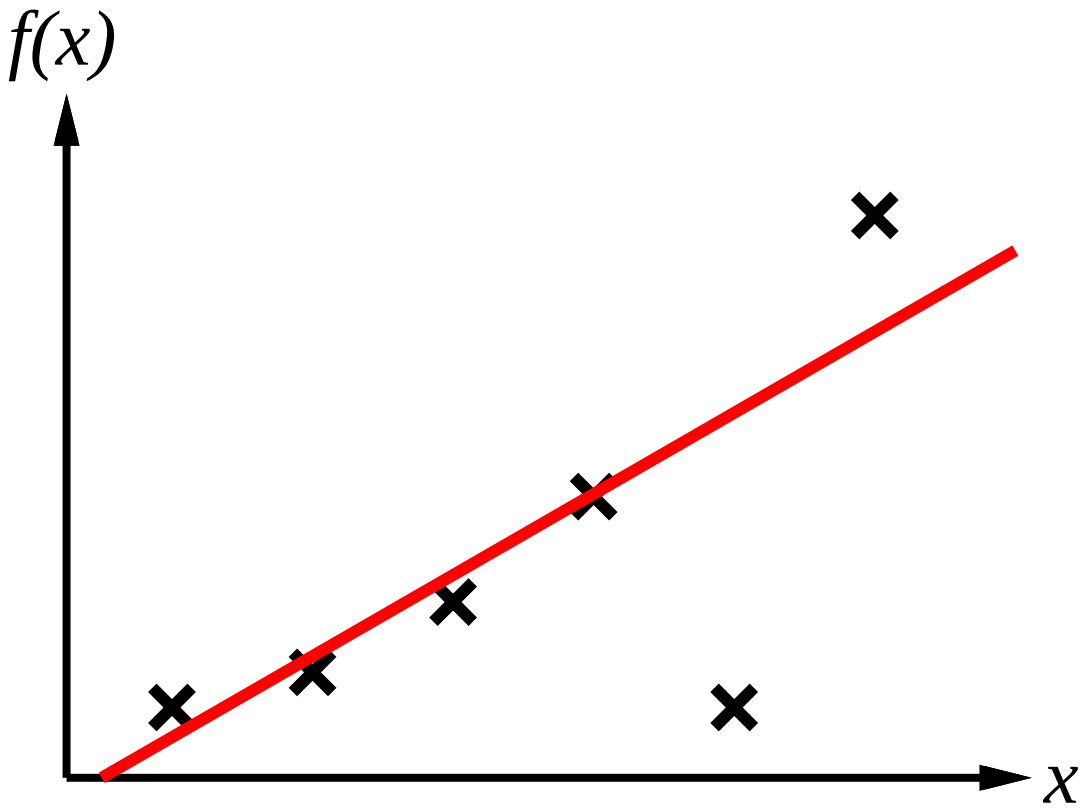
Searching dalam Supervised Learning

- Sebuah prosedur induktif mendefinisikan space: hypotheses space.
- Mis. untuk *curve-fitting*, hypotheses space = fungsi polynomial dgn. degree n:
  $f(x) = k_0 + k_1 x + k_2 x^2 + \ldots + k_n x^n$
- Search space terlalu kecil: $f(x)$ yang kita cari tidak ada (unrealisable)
- Search space terlalu besar:
  - Makin sulit ditelusuri
  - Makin banyak hipotesis yang konsisten dengan training example

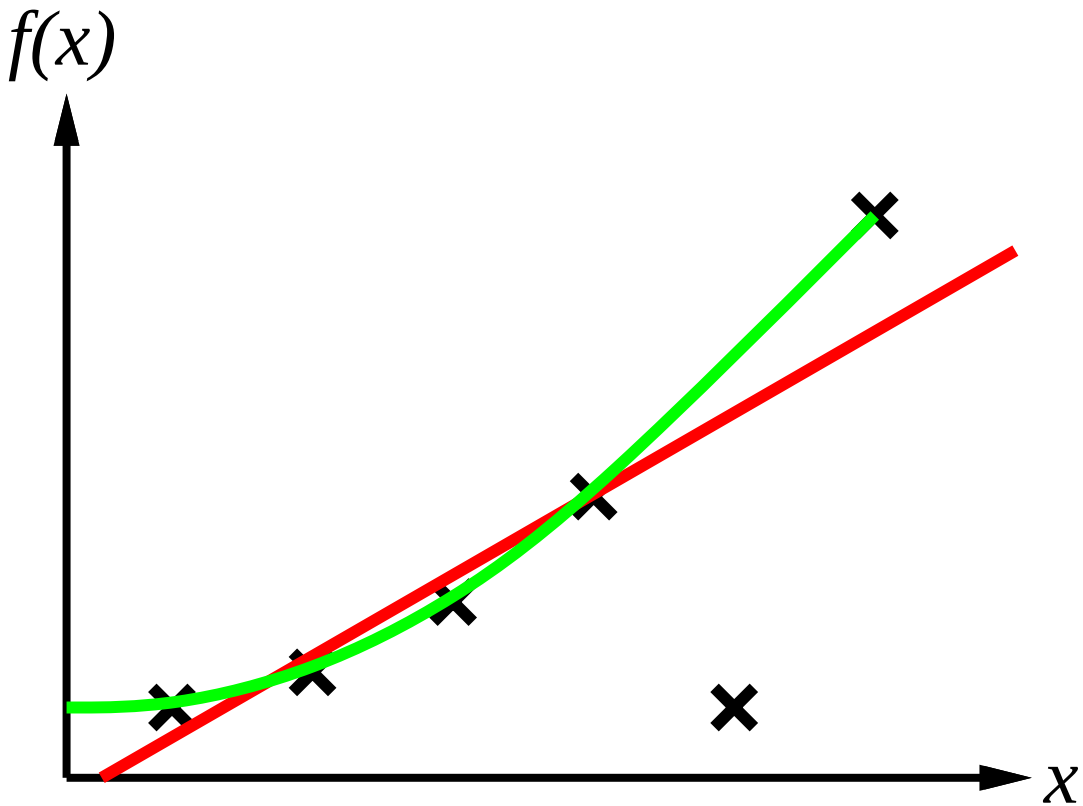
Contoh "curve-fitting"



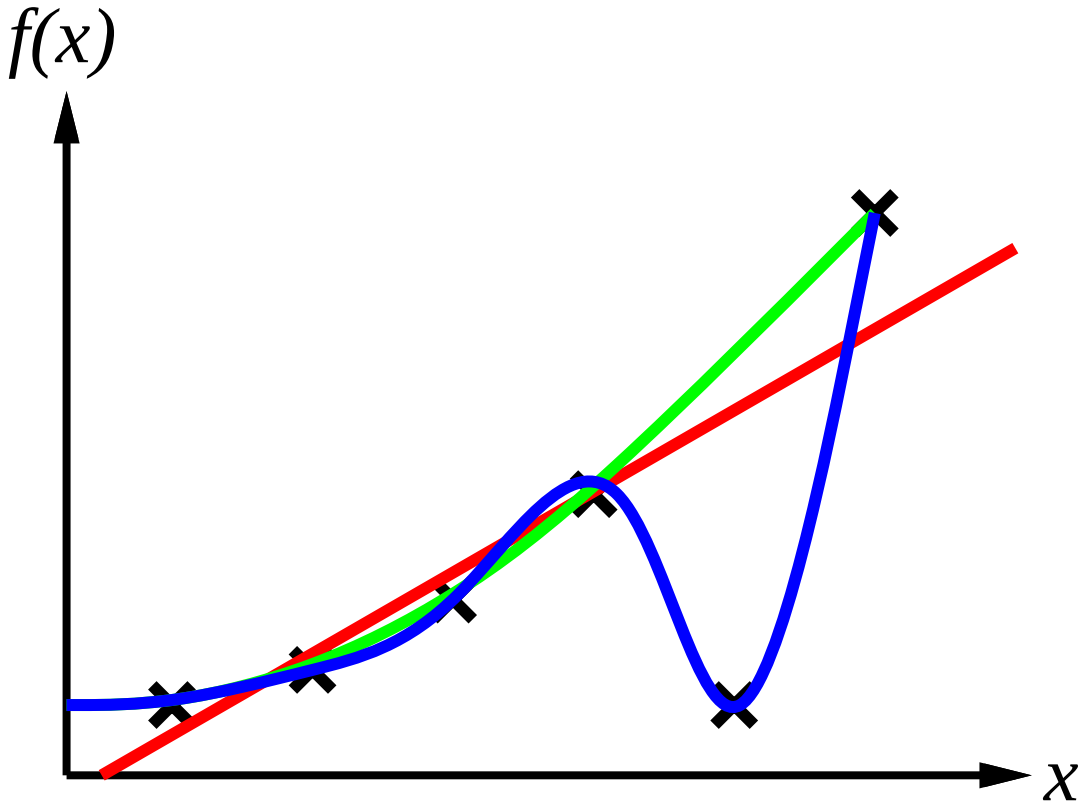
Contoh "curve-fitting"



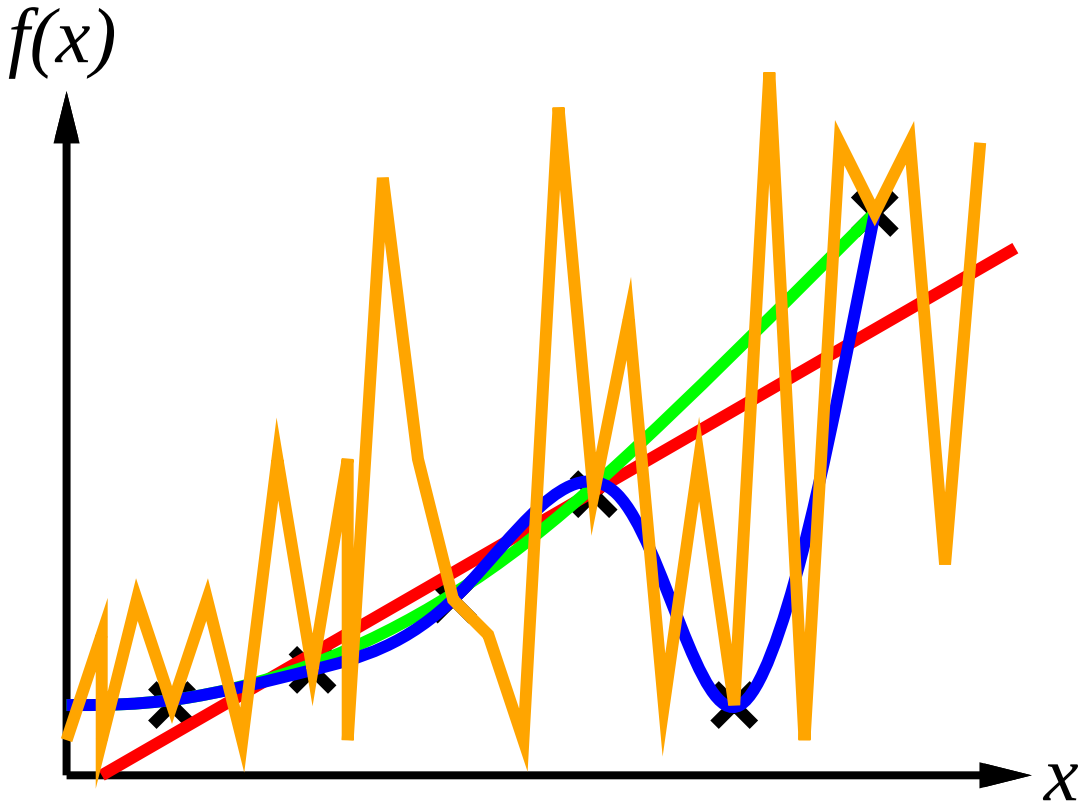
Contoh "curve-fitting"



Contoh "curve-fitting"



Contoh "curve-fitting"



Consistency vs. Simplicity

- Sebuah hipotesis yang *consistent* bisa menjelaskan semua *training example* (memetakan input ke output dengan akurat).
- Ada banyak consistent hypothesis untuk sebuah training set.
- Ockham's Razor: pilih yang paling *simple*! Secara intuitif: generalisasi terhadap example baru.
- Biasanya ada trade-off antara *consistency* dan *simplicity*.

Outline

Decision Tree Learning

- Decision tree merupakan representasi dari proses learning
- Hypothesis space mengandung himpunan *n* input variable dan 1 output variable.
- Tipe variable bisa: boolean, diskrit, kontinu
- Sebuah *training example* terdiri dari himpunan nilai input variable dan 1 output variable

Contoh: Menunggu di Restoran

SR berniat makan, berada di depan restoran, dan ingin memutuskan apakah rela menunggu/tidak untuk dapat meja di restoran.

Output variable:

WillWait (boolean): nilai *true* berarti menunggu

Cari metode yang dapat merumuskan fungsi hipotesis yang "menjawab" nilai *WillWait* untuk semua kemungkinan nilai input variable.

Contoh: Menunggu di Restoran

Input variable:

- *Alternate* (boolean): adakah restoran alternatif?
- *Bar* (boolean): apakah restoran memiliki bar?
- *Fri/Sat* (boolean): *true* jika hari ini adalah Jumat/Sabtu.
- *Patrons* (diskrit): ada berapa pengunjungnya? (None, Some, Full)
- *Price* (diskrit): Berapa kisaran harga hidangannya? ($, $$, $$$)
- *Raining* (boolean): apakah sedang hujan di luar?
- *Reservation* (boolean): apakah sudah membuat reservasi?
- *Type* (diskrit): apa jenis makanannya? (French, Italian, Thai, Burger)
- *WaitEstimate* (diskrit): Berapa lama perkiraan durasi menunggu? (0-10, 10-30, 30-60, >60 menit)

Contoh: Training Examples

Ex.	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
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*

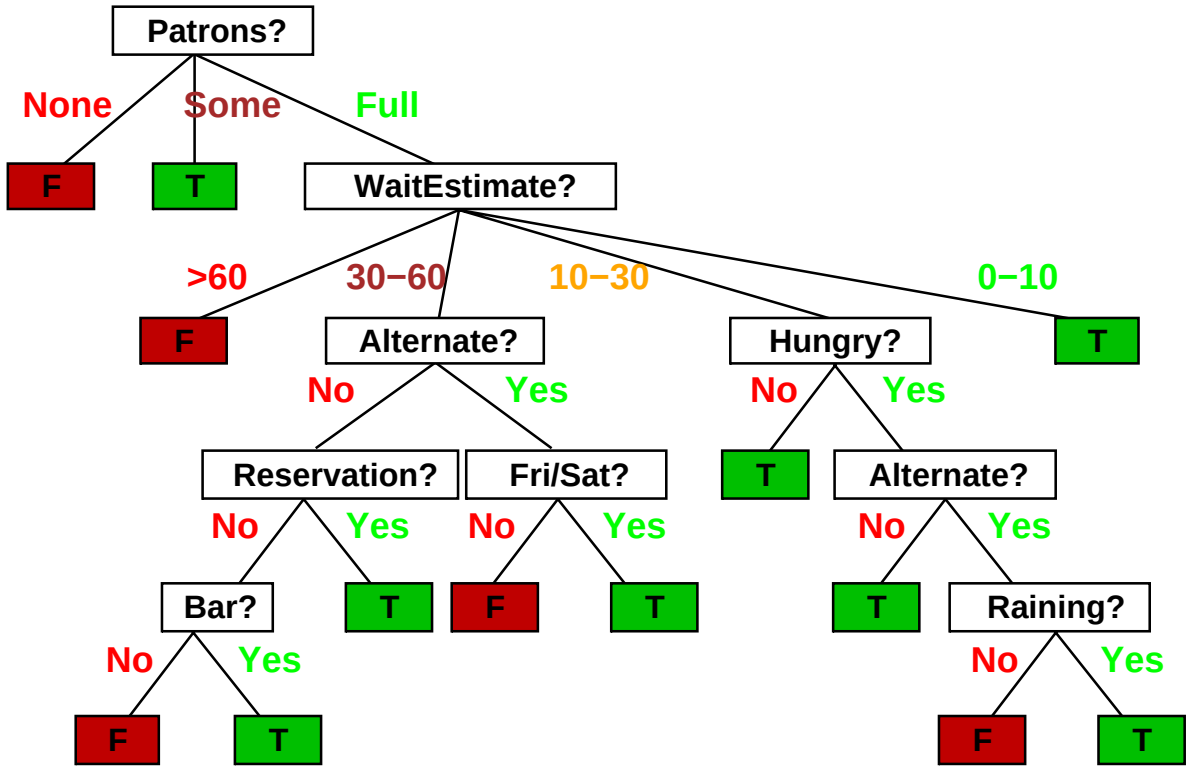
Boolean classification: Example di-classify menjadi positive (T) atau negative (F)

Decision Trees

- Sebuah representasi dari kemungkinan fungsi hipotesis
- Dapat dianggap sebagai sebuah **if...then** yang besar!
- Decision tree terdiri atas:
  - Internal node: representasi dari pengujian terhadap suatu nilai input variable
  - Himpunan edge dari suatu node: menyatakan semua kemungkinan nilai dari suatu input variable
  - Leaf node: memberikan nilai fungsi (output)

Decision Trees: Contoh

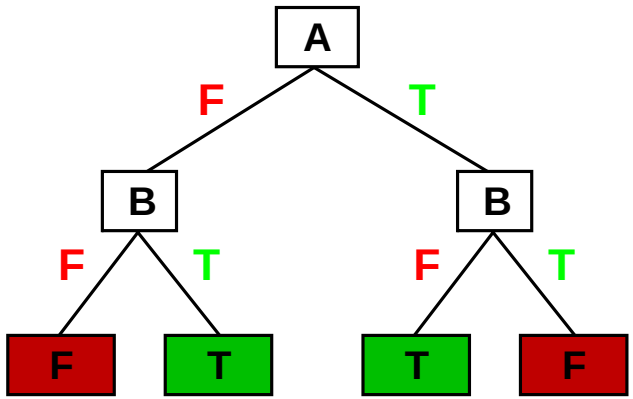
Mis.: inilah decision tree untuk fungsi yang "benar":



Decision Trees: Expresiveness

Sebuah decision tree dapat menyatakan sembarang fungsi dari nilai input (attributes)
Mis. untuk n variable boolean, buat tree dari truth table-nya:

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



Tree ini memiliki satu path root → leaf untuk setiap baris truth table.
Kesimpulan: buat satu path untuk setiap training example. Ini namanya "hafal mati"! (Generalisasi? Prediksi contoh baru? Belajar?)

Ingat Ockham's Razor:

Cari decision tree yang paling simple tapi consistent dengan data

Algoritma DTL

Tujuan: cari tree yang simple dan konsisten dengan training examples

Algoritma DTL

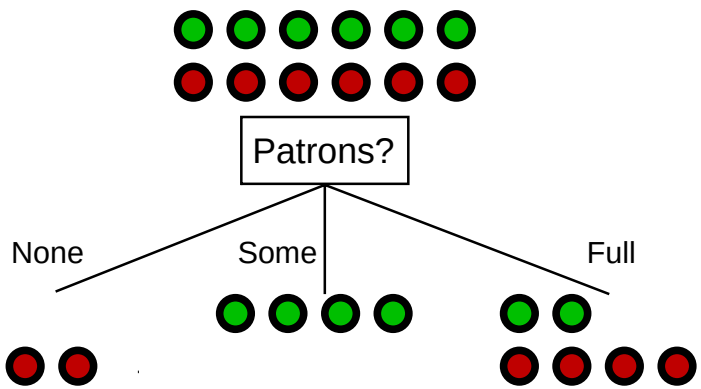
function DTL(*examples*, *attributes*, *parent_examples*) **returns** a tree

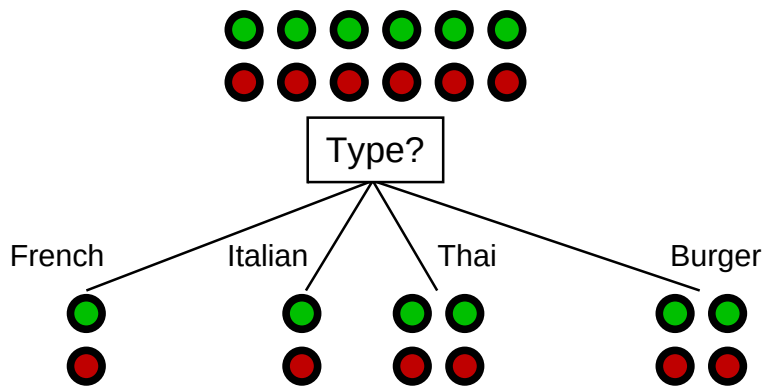
if *examples* is empty **then return** PLURALITY-VALUE(*parent_examples*)
else if all *examples* have the same classification **then return** classification
else if *attributes* is empty **then return** PLURALITY-VALUE(*examples*)
else
  $A \leftarrow \text{argmax}_{a \in attributes}$ IMPORTANCE(a, *examples*)
  tree $\leftarrow$ a new decision tree with root test A
  for each value v_k of A **do**
    exs $\leftarrow \{e : e \in examples$ **and** $e.A = v_k\}$
    subtree $\leftarrow$ DTL(*exs*, *attributes* $- A$, *examples*)
    add a branch to *tree* with label $(A = v_k)$ and subtree *subtree*
  return *tree*

Fungsi PLURALITY-VALUE memilih nilai output terbanyak dari himpunan example. Jika masing-masing output sama banyak, pilih acak.
Secara rekursif: cari input variable yang "paling menjelaskan" training example, tambahkan node-nya.

Memilih Variable

- implementasi dari fungsi IMPORTANCE pada algoritma DTL.
- Sebuah input variable yang ideal akan memilah example yang ada menjadi "semua positif" atau "semua negatif".
- Berdasarkan prinsip ini, sebuah variable bisa "lebih baik" dari variable lain.
- Contoh: mana yang lebih baik?





Information Theory: Entropy (1)

- Pilih variable yang paling banyak mengandung informasi mengenai nilai output variable.
- Gunakan konsep information gain dari Information Theory (Shannon & Weaver, 1949).
- Entropy adalah ukuran dari ketidakpastian informasi sebuah random variable.
- Bit: satuan informasi untuk menunjukkan besarnya informasi yang terkandung pada sebuah jawaban terhadap suatu pertanyaan.
  - Koin dengan kedua sisinya *heads* $= 0$ bit – tidak memiliki *ketidakpastian*
  - Koin dengan sisi *head* dan sisi *tail* $= 1$ bit $\rightarrow$ memiliki 2^1 nilai ($\langle 0.5, 0.5 \rangle$)
  - Dua koin yang memiliki *head* dan *tail* $= 2$ bit $\rightarrow$ memiliki 2^2 nilai ($\langle 0.25, 0.25, 0.25, 0.25 \rangle$)
- Berapa entropy untuk sebuah koin yang kemunculan *head*-nya memiliki probabilitas 0.99?

Information Theory: Entropy (2)

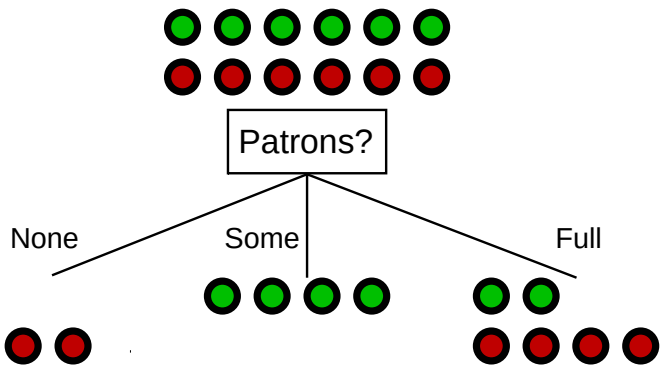
- Entropy dari random variable V, dengan nilai v_k dan $P(v_k)$ adalah probabilitas dari v_k:
  $H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)} = -\sum_k P(v_k) \log_2 P(v_k)$
- Dapat diperoleh definisi entropy untuk Boolean variable ($B(q)$) dimana q adalah probabilitas untuk nilai *true*:
  $B(q) = -(q \log_2 q + (1-q) \log_2 (1-q))$
- Jika training set tdd p data positif dan n data negatif, maka entropy dari atribut tujuan *Goal*:
  $H(Goal) = B(\frac{p}{p+n})$
- Untuk sebuah atribut A yang memiliki d nilai yang berbeda, maka training set E akan terbagi ke dalam subset: $E_1, \ldots, E_d$. Setiap E_k memiliki p_k data positif dan n_k data negatif:
  - Mengikuti percabangan ke-k: $B(p_k/(p_k + n_k))$ bits informasi.
  - Probabilitas memilih secara acak sample dari training set dengan atribut bernilai k: $(p_k + n_k)/(p + n)$.
  - *Expected entropy* tersisa setelah atribut A diketahui:
    $Remainder(A) = \sum_{k=1}^{d} \frac{p_k+n_k}{p+n} B(\frac{p_k}{p_k+n_k})$

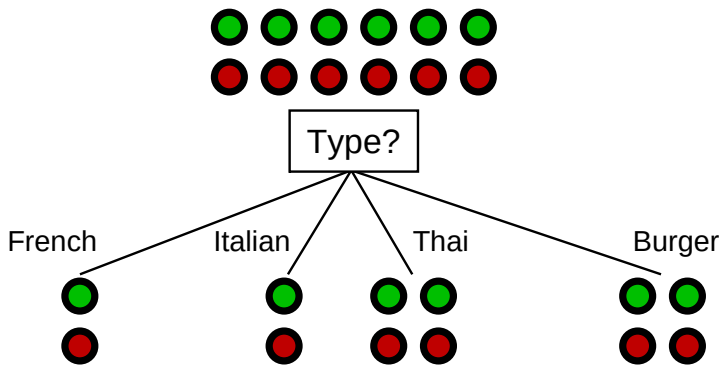
Information Theory: Information Gain

Information Gain

Selisih informasi yang dibutuhkan sebelum dan sesudah nilai sebuah atribut diketahui.

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





Mana yang lebih baik? Patrons? Type? Bagaimana information gain dari kedua atribut?

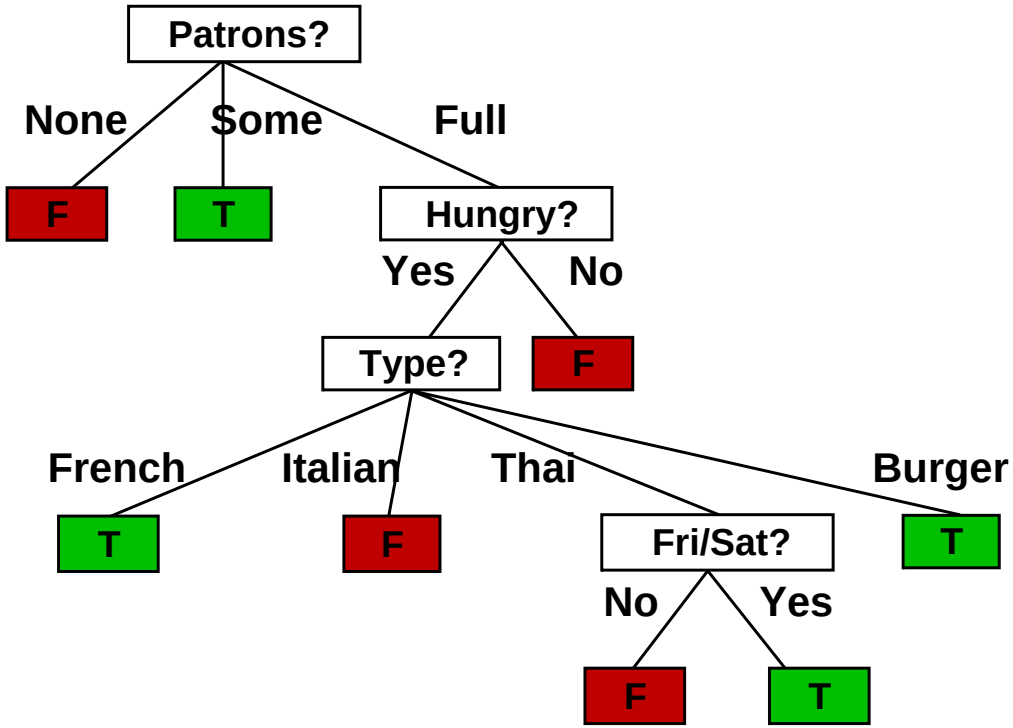
Memilih variable terbaik dengan Information Gain

Diketahui *outcome* dari *WillWait* ada 2 (*Yes* dan *No*), jumlah kasus dimana $WillWait = true$ adalah $p_1 = 6$, jumlah kasus dimana $WillWait = false$ adalah $p_2 = 6$.

- $H(WillWait) = H(\langle \frac{6}{6+6}, \frac{6}{6+6} \rangle) = H(\langle 0.5, 0.5 \rangle) = 1$
- $Remainder(Patrons) = \frac{2}{12} B(\frac{0}{2}) + \frac{4}{12} B(\frac{4}{4}) + \frac{6}{12} B(\frac{2}{6}) = 0.459$
- $Remainder(Type) = \frac{2}{12} B(\frac{1}{2}) + \frac{4}{12} B(\frac{2}{4}) + \frac{4}{12} B(\frac{2}{4}) + \frac{2}{12} B(\frac{1}{2}) = 1$
- $Gain(Patrons) = 1 - 0.459 = 0.541$
- $Gain(Type) = 1 - 1 = 0$

Hasil DTL

Tree yang dihasilkan algoritme DTL untuk 12 example:



- Lebih kecil/simple daripada tree yang *sebenarnya*!
- Hipotesis lebih kompleks (mis. *Bar*, *Rain*) tidak diperlukan.

Isu pada Decision Tree

- Jika ada atribut multivalue?
  Information gain tidak sesuai, gunakan gain ratio
- jika atribut pada input bernilai kontinue dan numerik?
  Cari split point yang memberikan Gain tertinggi (expensive!)
  - Urutkan data dari atribut ybs, ambil split point antara 2 example yang memiliki kelas berbeda, sambil hitung example masing-masing kelas pada kedua sisi dari split point.
- jika atribut pada output bernilai kontinu?
  Gunakan regression tree, bukan classification tree
  - *Regression tree* memiliki suatu fungsi linier pada setiap *leaf*, bukan suatu nilai.

Outline

Menguji Hipotesis (1)

- Bagaimana mengukur keberhasilan algoritme DTL dkk.?
- Uji kebenaran hipotesis "menjawab" example baru (generalisasi).
- Pendekatan sederhana: holdout cross-validation.
  - Bagi data secara random: training set dan test set
  - Jalankan learning pada training set
  - Evaluasi keberhasilan pada test set

Menguji Hipotesis (2)

Pendekatan lain:

- k-fold cross-validation
  - Bagi data menjadi k potongan/*fold* (1 potongan untuk test).
  - Train sebanyak k ronde; pada setiap ronde, gunakan potongan test set berbeda untuk validation.

Menguji Hipotesis (2)

Pendekatan lain:

- k-fold cross-validation
  - Bagi data menjadi k potongan/*fold* (1 potongan untuk test).
  - Train sebanyak k ronde; pada setiap ronde, gunakan potongan test set berbeda untuk validation.
  - Akurasi final adalah rerata dari akurasi k ronde.
- leave-one-out cross-validation (LOOCV)
  Seperti *k-fold* CV, namun pembagian data sebanyak elemen data ($k = n$).

Menguji Hipotesis (2)

Pendekatan lain:

- k-fold cross-validation
  - Bagi data menjadi k potongan/*fold* (1 potongan untuk test).
  - Train sebanyak k ronde; pada setiap ronde, gunakan potongan test set berbeda untuk validation.
  - Akurasi final adalah rerata dari akurasi k ronde.
- leave-one-out cross-validation (LOOCV)
  Seperti *k-fold* CV, namun pembagian data sebanyak elemen data ($k = n$).

Isu pada CV: *peeking*

- test set digunakan untuk memilih hipotesis sekaligus mengevaluasinya.

Menguji Hipotesis (2)

Pendekatan lain:

- k-fold cross-validation
  - Bagi data menjadi k potongan/*fold* (1 potongan untuk test).
  - Train sebanyak k ronde; pada setiap ronde, gunakan potongan test set berbeda untuk validation.
  - Akurasi final adalah rerata dari akurasi k ronde.
- leave-one-out cross-validation (LOOCV)
  Seperti *k-fold* CV, namun pembagian data sebanyak elemen data ($k = n$).

Isu pada CV: *peeking*

- test set digunakan untuk memilih hipotesis sekaligus mengevaluasinya.
- Pisahkan test set, kemudian baru membagi data menjadi training set dan validation set.

Learning Curve

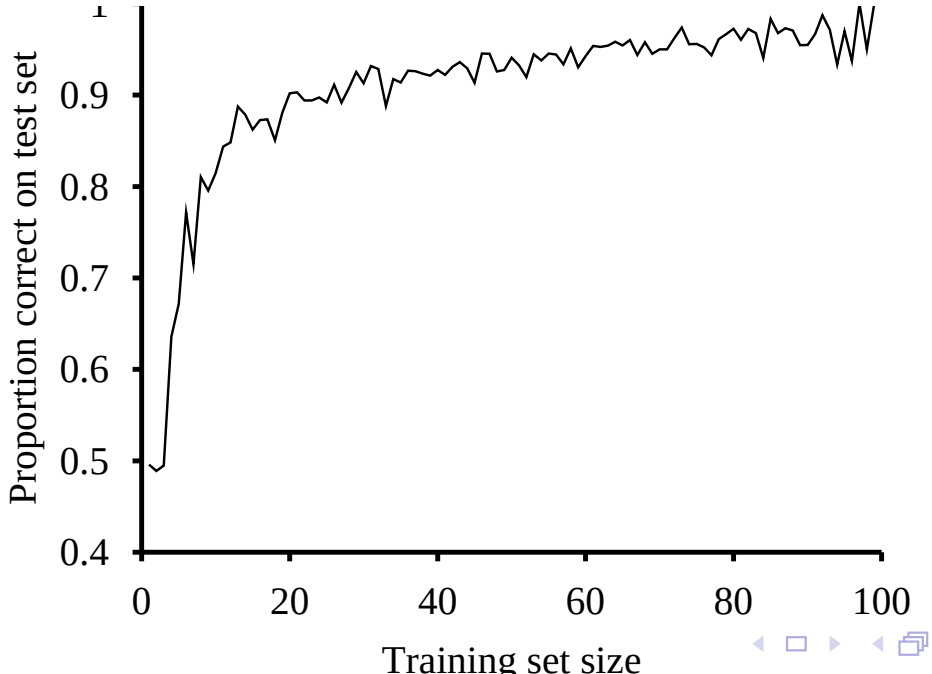
Knowledge = Power

Semakin banyak data, semakin bagus hasil machine learning.

Learning curve

% prediksi benar pada *test set* sbg. fungsi dari ukuran *training set*.

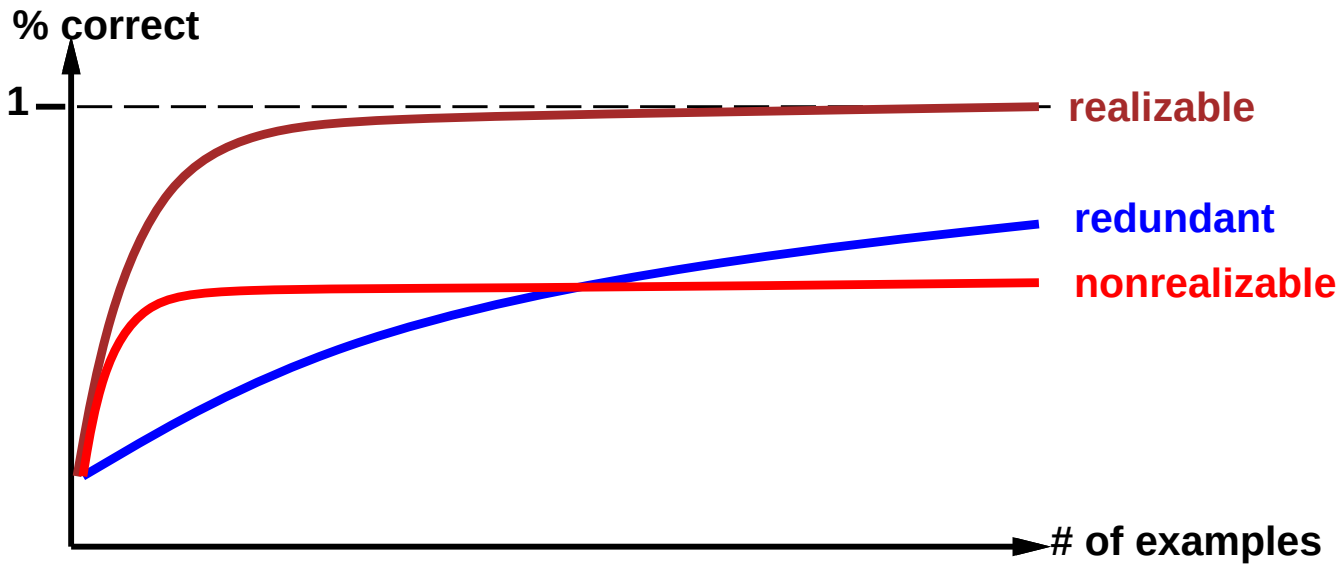
Contoh, pada data restoran:



Bentuk Learning Curve

Learning curve juga tergantung masalah yang dipelajari:

- Realizable: Fungsi target $f(x)$ bisa dinyatakan
- Non-realizable: Fungsi target $f(x)$ tidak bisa dinyatakan (kurang atribut?)
- Redundant: Banyak atribut *noise* yang tidak berguna, menyesatkan (overfitting)!



Ringkasan: Machine Learning

- Learning bermanfaat untuk:
  - Unknown environment
  - Lazy designers
- Supervised learning menggunakan prinsip induksi: dari sehimpunan data, estimasi sebuah hipotesis
- *Trade-off* antara consistency dan simplicity
- Algoritma Decision Tree Learning menggunakan Information Gain
- Metode machine learning diuji dengan tahap training dan testing
- CSCE604235 Pemelajaran Mesin (*Machine Learning*)