IF5181 Pengenalan Pola

Klasifikasi

Masayu Leylia Khodra

Referensi: Bab 8 dari Han, J., Pei, J., & Kamber, M. (2011). Data mining: concepts and techniques. Elsevier.

Outline

- Klasifikasi: what, how
- Klasifikasi berbasis pengetahuan
- Klasifikasi berbasis pembelajaran mesin
  - Decision tree learning
  - Naive Bayes
  - IF-THEN rules
    - Ekstraksi dari Decision Tree
    - Rule Induction: Sequential Covering Method
  - Neural Network

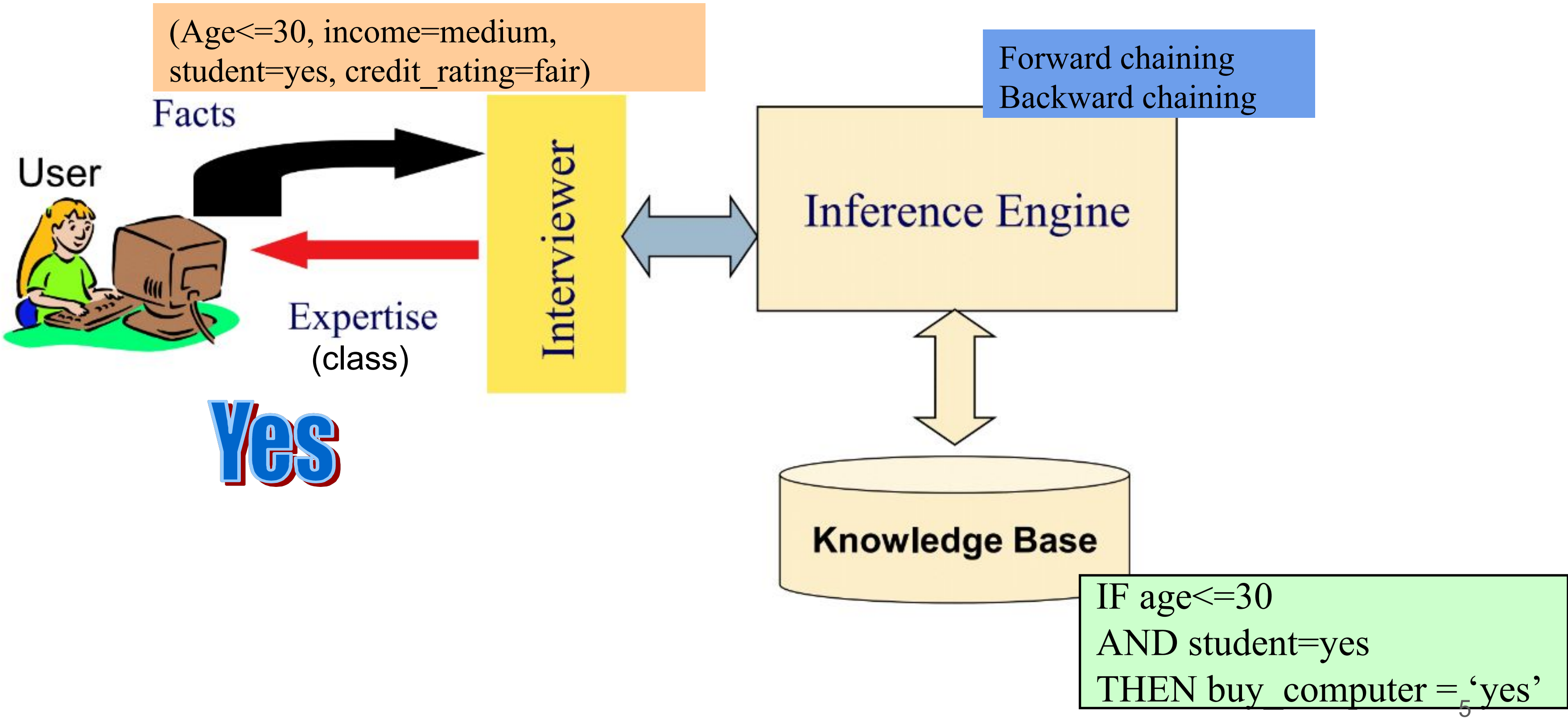
Klasifikasi: Aplikasi

- Traffic sign recognition
- Kategorisasi aspek
- Klasifikasi sentimen
- Identifikasi kalimat majemuk
- Klasifikasi intent dari kalimat
- Spam filtering
- Klasifikasi citra hiu atau bukan
- POS Tagging
- Named-entity recognition
- Diagnosis penyakit
- Identifikasi pesawat tempur

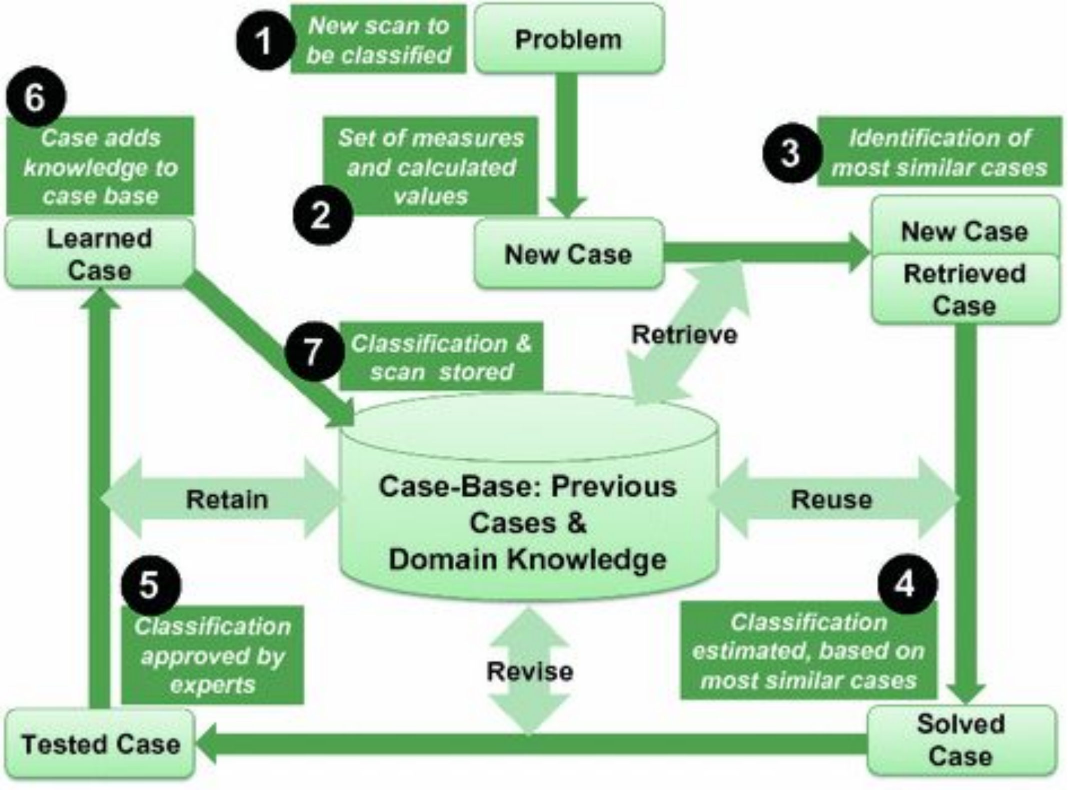
Klasifikasi

- Klasifikasi: proses menentukan kategori yang diambil dari predefined-category set.
- Fungsi Klasifikasi: data → class ∈ C
- Pendekatan (Sebastiani, 2005; Aly, 2005):
  - Berbasis pengetahuan
  - Berbasis pembelajaran mesin: producing a learning model from a labeled training set

Klasifikasi berbasis Pengetahuan



Cased-based Classification

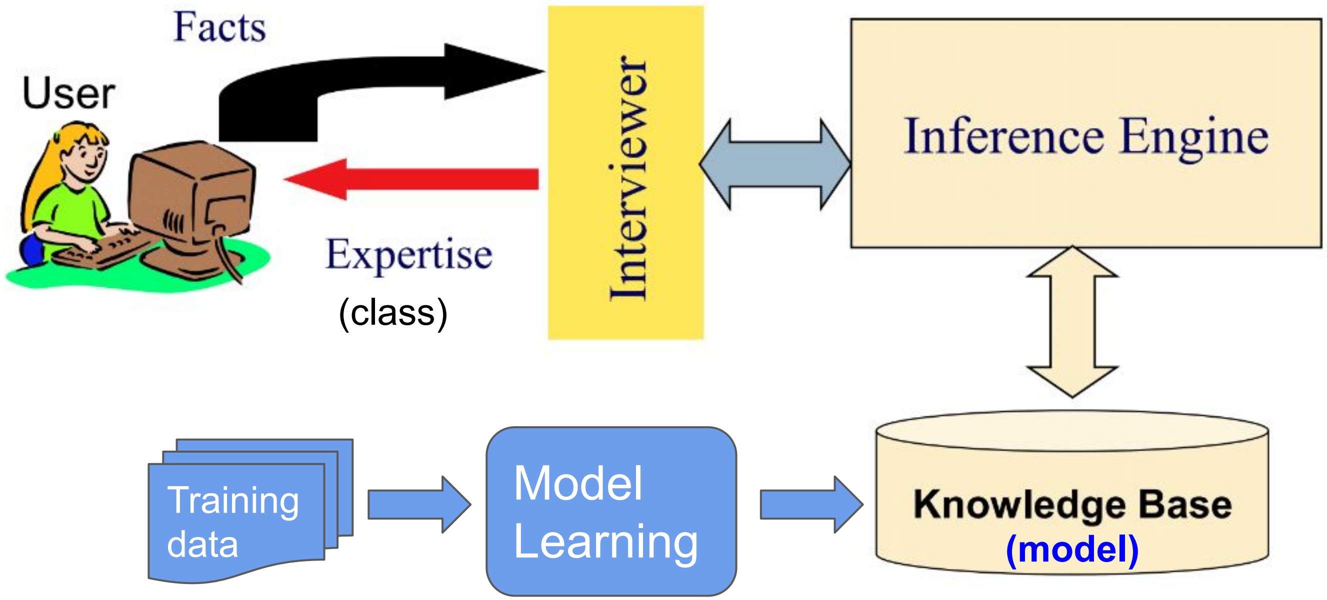


- Case-based reasoning:
  - Retrieve
  - Reuse
  - Revise
  - Retain

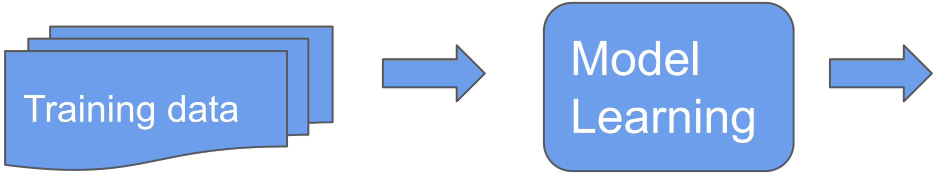
Klasifikasi berbasis Pembelajaran Mesin

- Konstruksi model:
  - Training data: kumpulan data berlabel untuk konstruksi model.
  - Model direpresentasikan sebagai aturan klasifikasi, pohon keputusan, atau representasi pengetahuan lainnya.
- Validasi dan Tes model: estimasi kinerja model terhadap test data.
  - Test data harus independent dari training data.
  - Test data untuk memilih model disebut validation data.
- Model deployment: klasifikasi data baru

Konstruksi Model



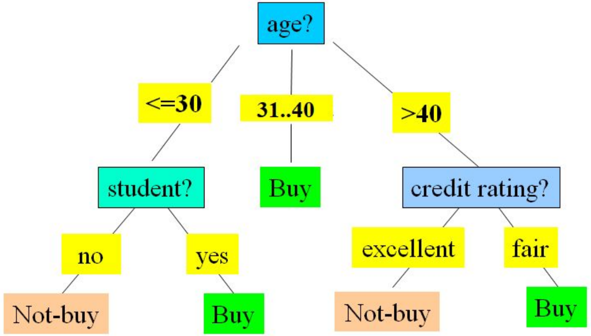
Konstruksi Model: Contoh



age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Decision tree learning

Naive Bayes learning

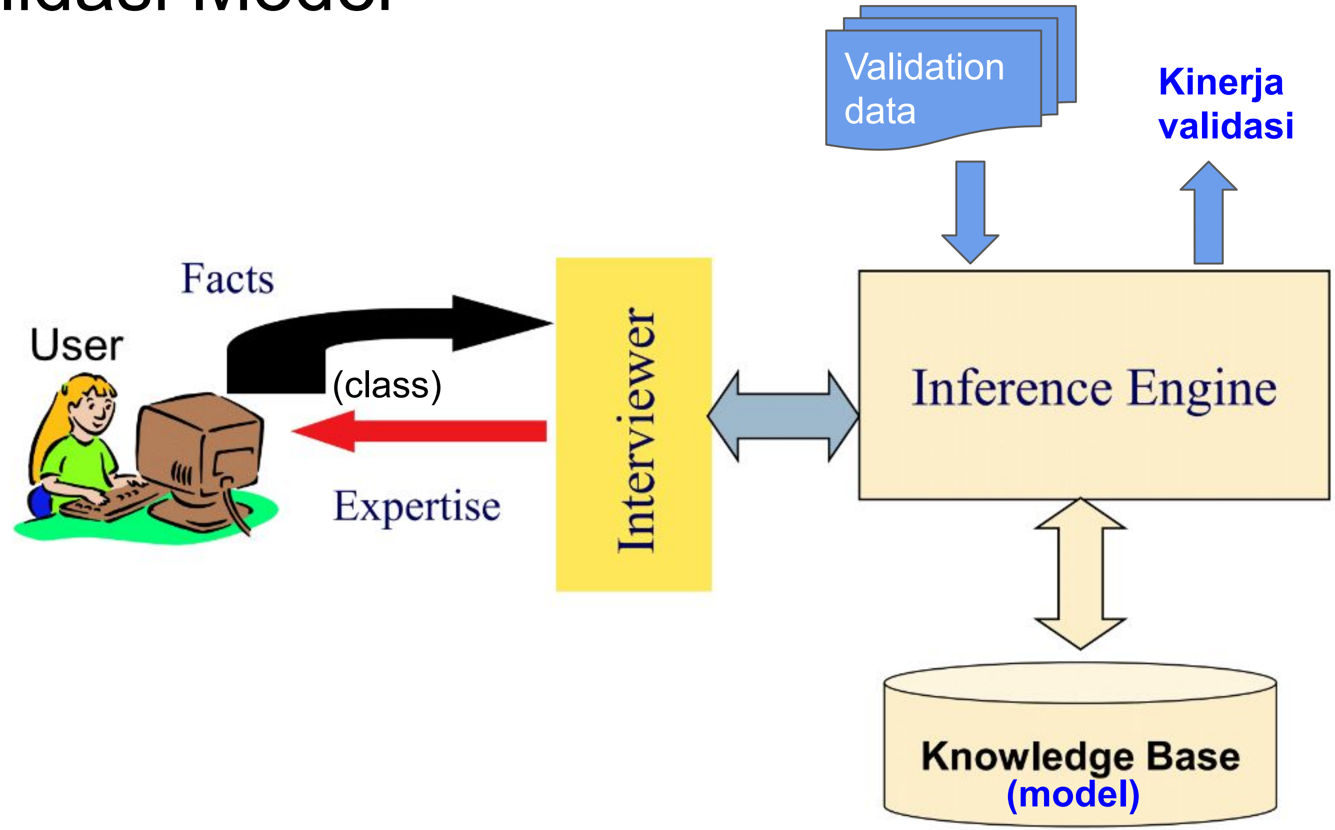


P(buys_computer = “yes”) = 9/14 = 0.643

P(buys_computer = “no”) = 5/14= 0.357

P(age = “<=30”|buys_computer = “yes”) = 2/9 = 0.222
P(age = “<= 30”|buys_computer = “no”) = 3/5 = 0.6
P(income = “medium” | buys_computer = “yes”) = 4/9 = 0.444
P(income = “medium” | buys_computer = “no”) = 2/5 = 0.4
P(student = “yes” | buys_computer = “yes) = 6/9 = 0.667
P(student = “yes” | buys_computer = “no”) = 1/5 = 0.2
P(credit_rating = “fair” | buys_computer = “yes”) = 6/9 = 0.667
P(credit_rating = “fair” | buys_computer = “no”) = 2/5 = 0.4

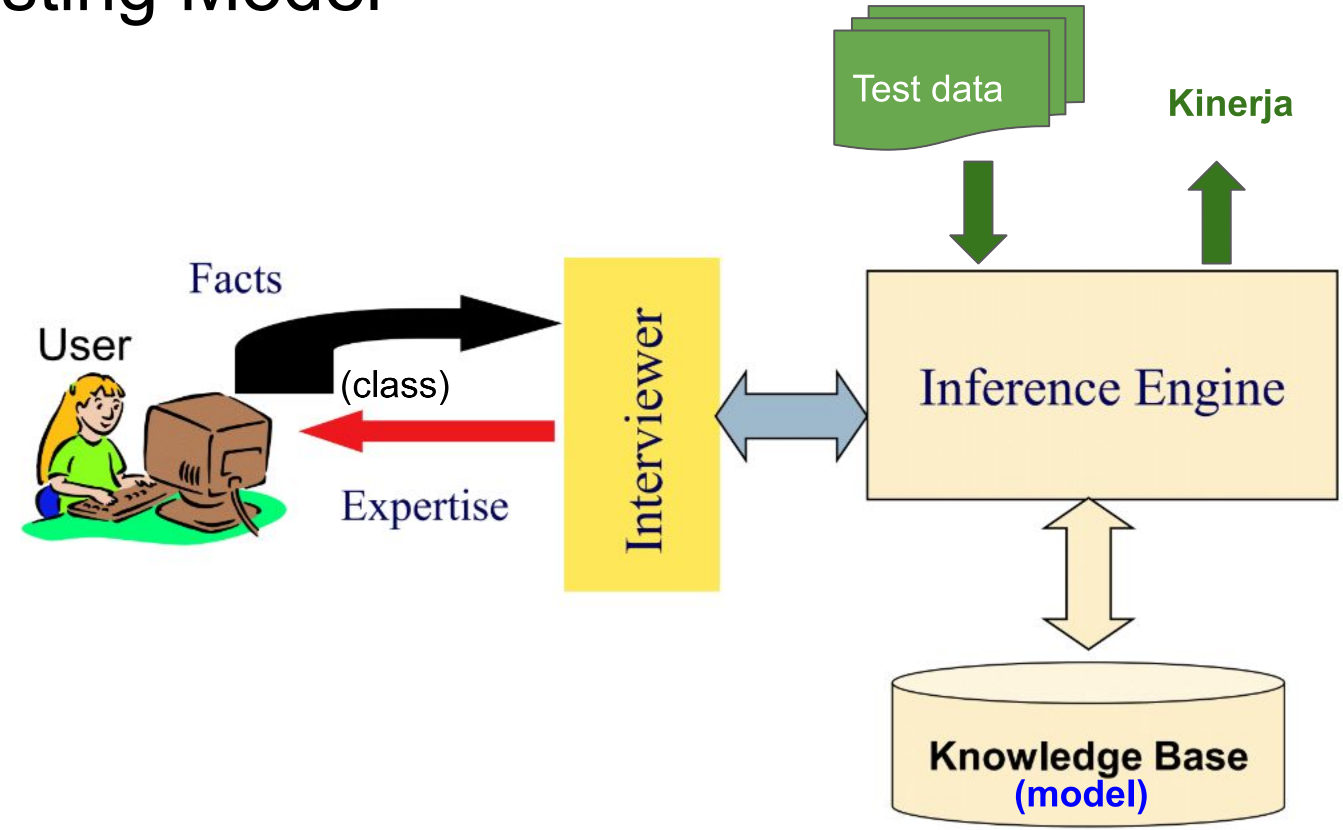
Validasi Model



Model	Kinerja validasi
M1	K1
...	...
Mn	Kn

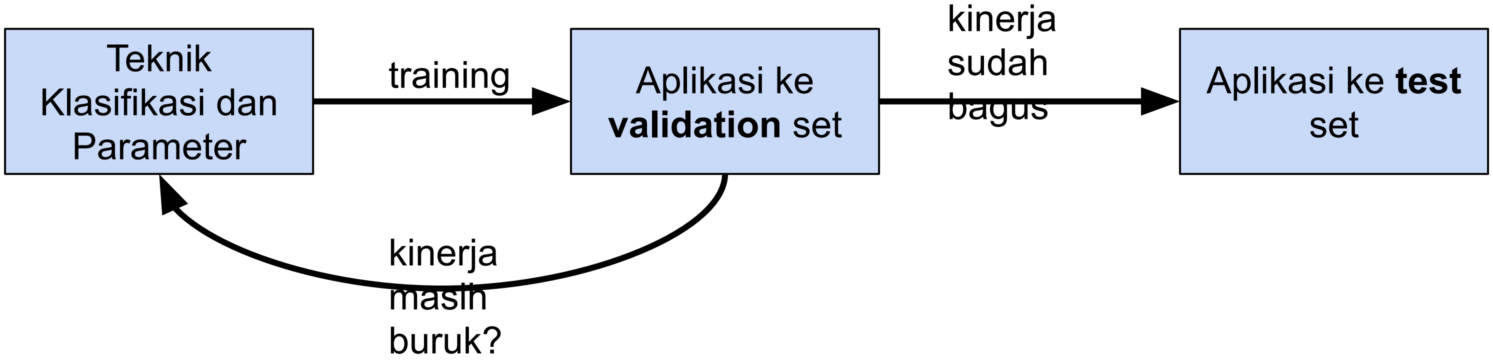
Model terbaik

Testing Model



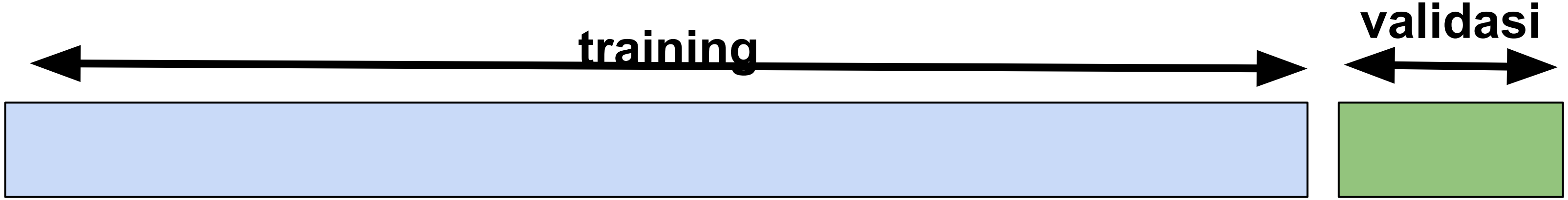
Validasi & Tes Model

- Idealnya ada tiga jenis dataset: training set, validation set, test set.
- Kinerja model = kinerja pada test-set.
- Test-set hanya digunakan satu kali.

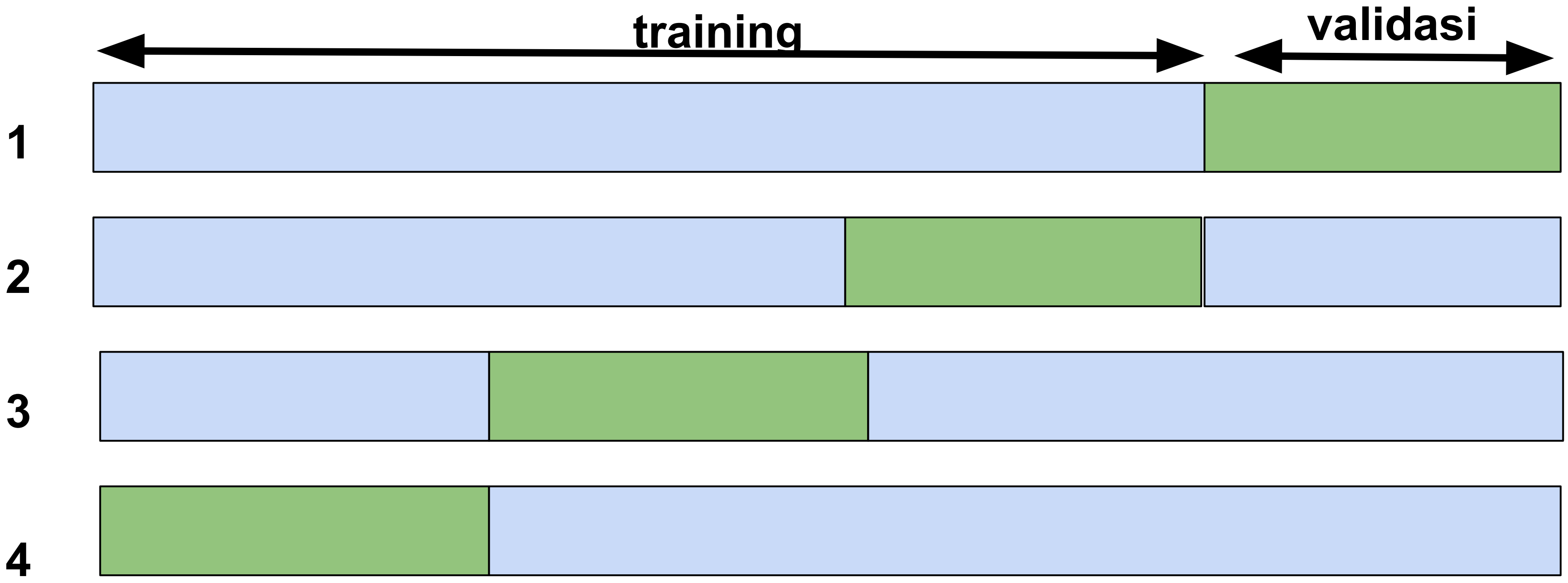


Hold-out Validation

- Salah satu cara: pisahkan dataset menjadi training data dan validation data. Umumnya: 80% training, 20% validation data (atau 90:10)



K-Fold Cross Validation



Evaluasi: Confusion Matrix

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Pengukuran

		Predicted class	
		+ve	-ve
Actual Class	+ve	True Positive(TP)	False Negative(FN)
	-ve	False Positive(FP)	True Negative(TN)

Akurasi = $\frac{TP+TN}{TP+FP+TN+FN}$

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.460.8051&rep=rep1&type=pdf

F-Measure

		Predicted class	
		+ve	-ve
Actual Class	+ve	True Positive(TP)	False Negative(FN)
	-ve	False Positive(FP)	True Negative(TN)

$$Precision = \frac{X}{Y} \texttt{ or } \frac{TP}{TP + FP}$$

$$Recall = \frac{Y}{Z} \texttt{ or } \frac{TP}{TP + FN}$$

Precision: % item yang dipilih yang benar.

Recall: % item benar yang diambil.

F-Measure

F-Measure: kombinasi antara nilai prec. dan recall

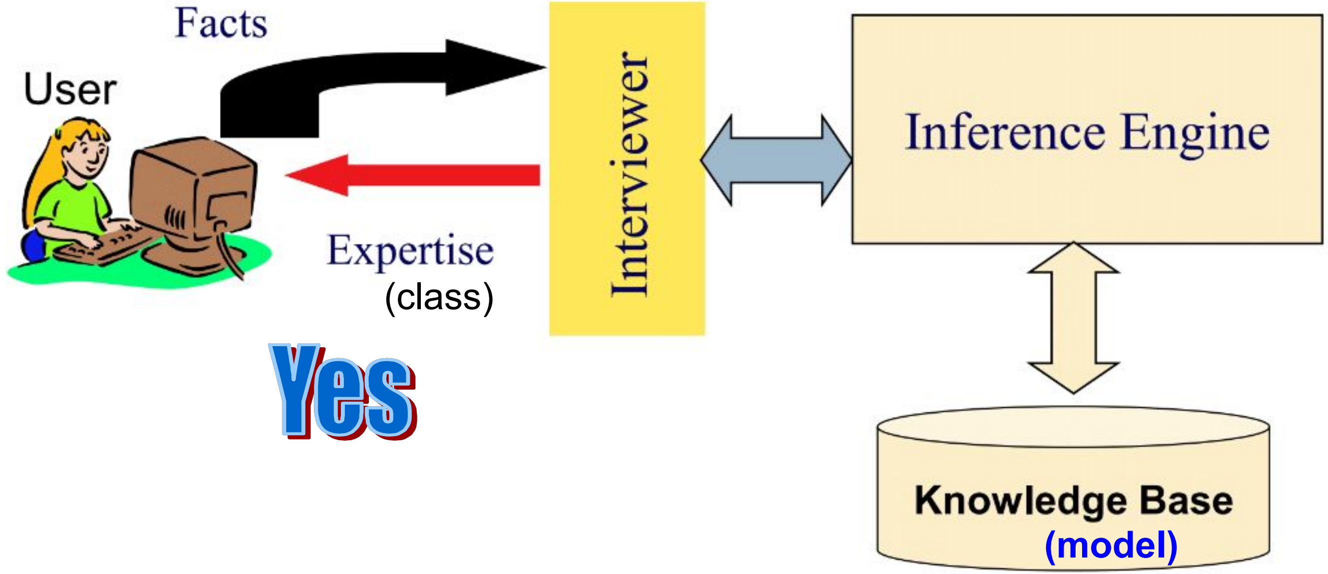
$$F - measure = \frac{2 \times Recall \times Precision}{Recall + Precision}$$

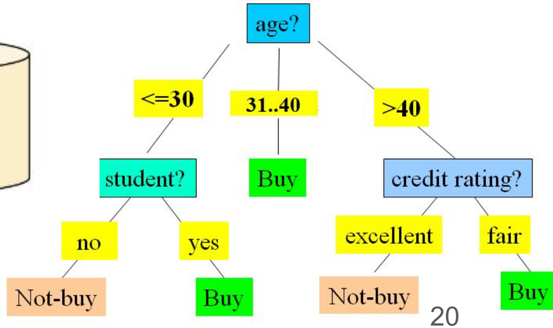
Ukuran Kinerja Lain

- Kappa (Cohen’s kappa):
  - akurasi dinormalisasi dengan imbalance dari class.
- ROC Curves
- Geometric mean of the true rates
- Dominance

Model Deployment

(Age<=30, income=medium,
student=yes, credit_rating=fair)





Review Tahapan Pembelajaran Mesin

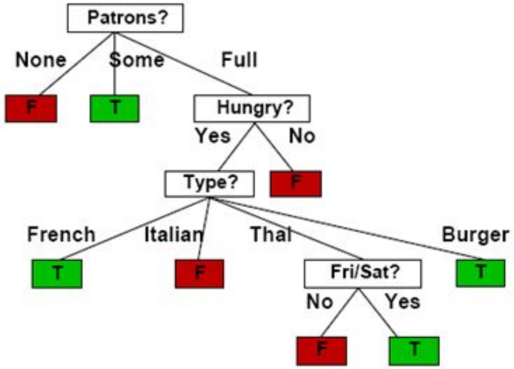
- Proses konstruksi model
  - Training data → Model Klasifikasi
- Proses validasi dan tes model
  - Validation data → model terbaik
  - Test data → kinerja model
- Proses klasifikasi (deployment)
  - Data yang tidak diketahui kelasnya → kelas data

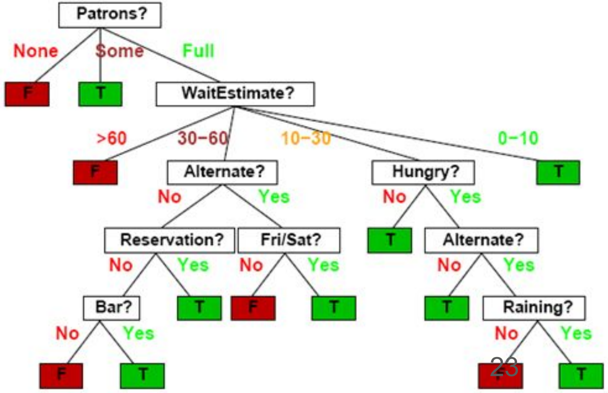
Algoritma Pembelajaran

- Decision tree learning
- Naive Bayes
- IF-THEN rules
  - Ekstraksi dari Decision Tree
  - Rule Induction: Sequential Covering Method
- Neural Network

Decision Tree Learning

Example	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





Russell, S. J., & Norvig, P. (2016). *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,.

Decision Tree Learning

- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose "most significant" attribute as root of

```
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
```

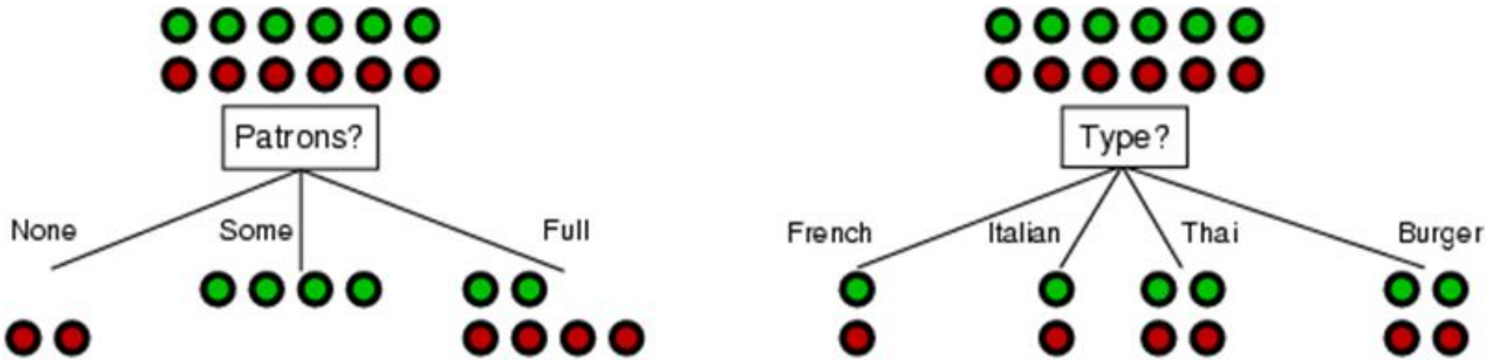
$$Info(D) = -\sum_{i=1}^{m} p_i \log_2(p_i)$$

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j)$$

$$Gain(A) = Info(D) - Info_A(D)$$

Choosing an attribute

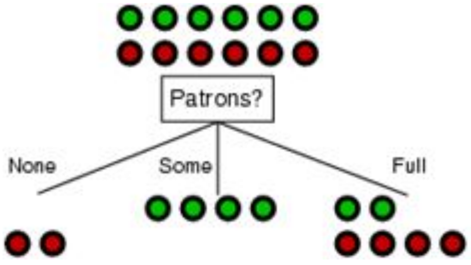
Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



$$IG(Patrons) = 1 - [\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I(\frac{2}{6}, \frac{4}{6})] = 0.541 \text{bits}$$

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

Patrons? = Full



Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

I(1/3,2/3)=0.92

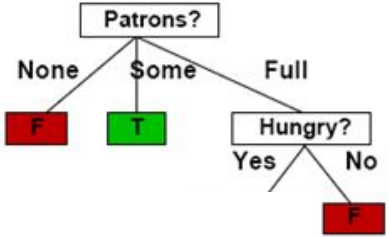
IG(Alt)=0.92-5/6*I(2/5,3/5)-1/6*I(0,1)=0.11

...

IG(Hun)=0.92-4/6*I(1/2,1/2)-2/6*I(0,1)=0.25

...

Patrons? = Full and Hungry? = Yes



Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*\$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*\$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*\$\$\$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*\$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

I(1/2,1/2)=1

IG(Alt)=1-4/4*I(1/2,1/2)=0

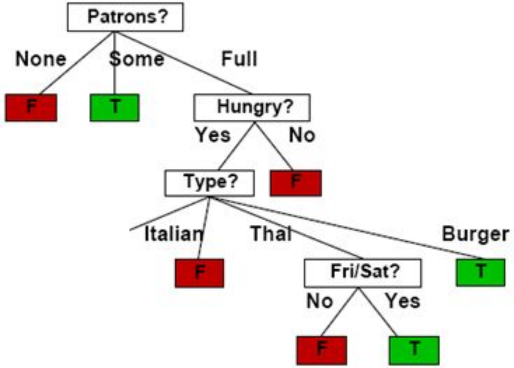
IG(Bar)=1-1/2*I(1/2,1/2)-1/2*I(1/2,1/2)=0

IG(Fri)=1-1/4*I(0,1)-3/4*I(2/3,1/3)=0.39

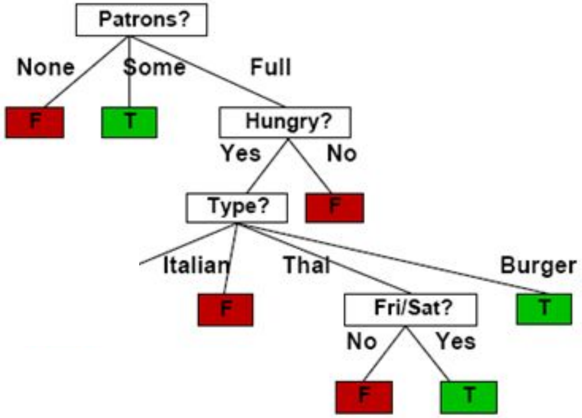
...

IG(Type)=1-1/2*I(1/2,1/2)-1/4*I(0,1)-1/4*I(1,0)=0.5

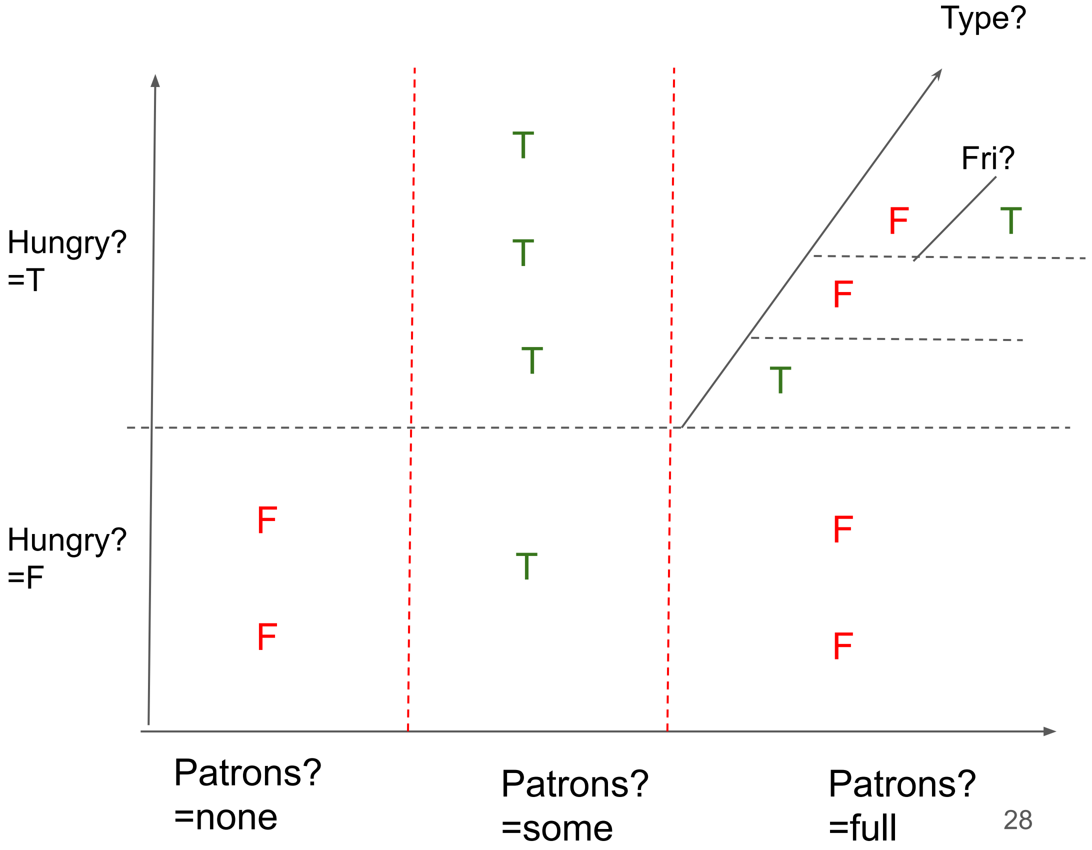
...



Klasifikasi dengan Hyperplane Decision Tree



Example	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



Decision Tree Learning: Why Popular

- Relatively fast learning speed
- Convertible to simple and easy to understand classification rules
- Easy to be adapted to database system implementations (e.g., using SQL)
- Comparable classification accuracy with other methods

Naive Bayes Classifier

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

P(buys_computer = “yes”) = 9/14 = 0.643

P(buys_computer = “no”) = 5/14= 0.357

P(age = “<=30”|buys_computer = “yes”) = 2/9 = 0.222

P(age = “<= 30”|buys_computer = “no”) = 3/5 = 0.6

P(income = “medium” | buys_computer = “yes”) = 4/9 = 0.444

P(income = “medium” | buys_computer = “no”) = 2/5 = 0.4

P(student = “yes” | buys_computer = “yes) = 6/9 = 0.667

P(student = “yes” | buys_computer = “no”) = 1/5 = 0.2

P(credit_rating = “fair” | buys_computer = “yes”) = 6/9 = 0.667

P(credit_rating = “fair” | buys_computer = “no”) = 2/5 = 0.4

Naive Bayes: Learning

P(age = “<=30”|buys_computer = “yes”)
= 2/9 = 0.222

P(age = “<= 30”|buys_computer = “no”)
= 3/5 = 0.6

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31...40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31...40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31...40	medium	no	excellent	yes
31...40	high	yes	fair	yes
>40	medium	no	excellent	no

$$v_{NB} = \underset{v_j \in \{yes,no\}}{\arg\max}\, P(v_j)\prod_i P(a_i \mid v_j)$$

$P(v_j)$: probabilitas kelas v_j

$P(a_i|v_j)$: probabilitas atribut a_i pada v_j

$P(a_i|v_j) = (v_j \cap a_i) / |v_j|$

Naive Bayes: Klasifikasi

$$v_{NB} = \underset{v_j \in \{yes, no\}}{\arg\max} P(v_j) \prod_i P(a_i \mid v_j)$$

P(buys_computer = “yes”) = 9/14 = 0.643

P(buys_computer = “no”) = 5/14= 0.357

P(age = “<=30” | buys_computer = “yes”) = 2/9 = 0.222

P(age = “<= 30” | buys_computer = “no”) = 3/5 = 0.6

P(income = “medium” | buys_computer = “yes”) = 4/9 = 0.444

P(income = “medium” | buys_computer = “no”) = 2/5 = 0.4

P(student = “yes” | buys_computer = “yes) = 6/9 = 0.667

P(student = “yes” | buys_computer = “no”) = 1/5 = 0.2

P(credit_rating = “fair” | buys_computer = “yes”) = 6/9 = 0.667

P(credit_rating = “fair” | buys_computer = “no”) = 2/5 = 0.4

X = (age <= 30 , income = medium, student = yes, credit_rating = fair)

P(X|buys_computer = “yes”) = 0.222 x 0.444 x 0.667 x 0.667 = 0.044
P(X|buys_computer = “yes”) * P(buys_computer = “yes”) = 0.028

P(X|buys_computer = “no”) = 0.6 x 0.4 x 0.2 x 0.4 = 0.019
P(X|buys_computer = “no”) * P(buys_computer = “no”) = 0.007

Naive Bayes: Keuntungan vs Kerugian

- Keuntungan
  - Mudah untuk dibuat
  - Hasil bagus
  - Incremental learning
- Kerugian
  - Asumsi independence antar atribut

Klasifikasi dengan IF-THEN rules

age	income	student	credit_rating	buys_computer
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

IF age = young AND student = no THEN buys_computer = no

IF age = young AND student = yes THEN buys_computer = yes

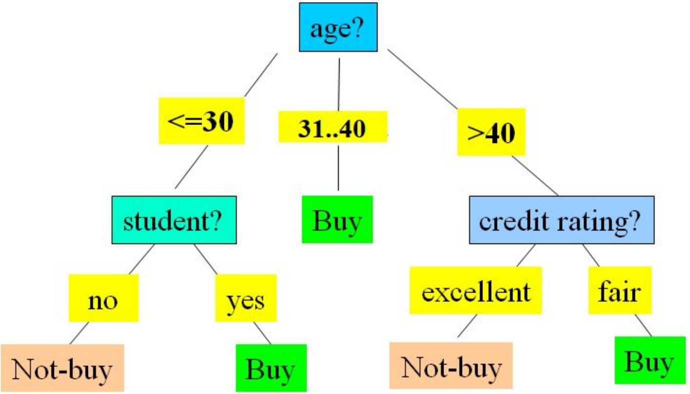
IF age = mid-age THEN buys_computer = yes

IF age = old AND credit_rating = excellent THEN buys_computer = no

IF age = old AND credit_rating = fair THEN buys_computer = yes

Klasifikasi dengan IF-THEN rules (2)

- Represent the knowledge in the form of IF-THEN rules
  Format: IF rule-precondition THEN rule-consequent
  Contoh: IF age = youth AND student = yes THEN buys_computer = yes
- Ekstraksi rule dari decision trees karena lebih mudah dimengerti daripada trees.



IF age = young AND student = no THEN buys_computer = no

IF age = young AND student = yes THEN buys_computer = yes

IF age = mid-age THEN buys_computer = yes

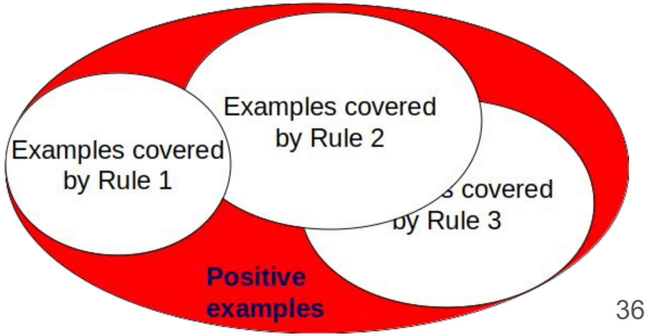
IF age = old AND credit_rating = excellent THEN buys_computer = no

IF age = old AND credit_rating = fair THEN buys_computer = yes

Rule Induction: Sequential Covering Algorithm

- Sequential covering algorithm: Extracts rules directly from training data
- Rules are learned sequentially, each for a given class Ci will cover many tuples of Ci but none (or few) of the tuples of other classes

while (enough target tuples left)
    generate a rule
    remove positive target tuples satisfying this rule



Sequential Covering Algorithm

COVER(*Target_attr, Attrs, Examples, Threshold*)

- *Learned_rules* ← {}
- *Rule* ← LEARN-ONE-RULE(*Target_attr, Attrs, Examples*)
- WHILE PERFORMANCE(*Rule, Examples*) > *Threshold*, DO
  - *Learned_rules* ← *Learned_rules* + *Rule*
  - *Examples* ← *Examples* − {EXAMPLES CORRECTLY CLASSIFIED BY *Rule*}
  - *Rule* ← LEARN-ONE-RULE(*Target_attr, Attrs, Examples*)
- *Learned_rules* ← SORT *Learned_rules* ACCORD TO PERFORMANCE OVER *Examples*
- RETURN *Learned_rules*

- Algoritma covering: bangkitkan rules yang meng-cover kelas spesifik
- Steps:
  - Generate rule R on training data
  - Remove training data covered by R
  - Repeat the process

http://www.cs.cornell.edu/courses/cs478/2000SP/lectures/rule-learning.pdf

Learn-One-Rule

LEARN-ONE-RULE($Target\_attr$, $Attrs$, $Examples$)

- $Pos \leftarrow$ positive $Examples$; $Neg \leftarrow$ negative $Examples$
- If Pos

  $NewRule \leftarrow$ most general rule possible; $NewRuleNeg \leftarrow Neg$

  While $NewRuleNeg$

  1. $Candidate\_literals(CLs) \leftarrow$ generate candidates
  2. $Best\_literal \leftarrow argmax_{L \in CLs}$ $PERFORMANCE(Specialize(NewRule, L))$
  3. add $Best\_literal$ to $NewRule$ preconditions
  4. $NewRuleNeg \leftarrow$ subset of $NewRuleNeg$ that satisfies $NewRule$ preconditions
- Return $NewRule$

T: attributes

C: classes

For all A in T do:

  For all possible value vi in A do:

    For all Cj in C do:

      find count(Cj)

    Cmax=class with largest count

    RA=RA ∪ (A=v ⇒ Cmax)

  ErrA=number of incorrectly classified by RA

R=RA where ErrA is minimum

https://www.slideshare.net/totoyou/covering-rulesbased-algorithm

Learn-One-Rule: Contoh

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*Ski?*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Attribute	Rules	Error	Total
Outlook	outlook=Sunny⇒No	2/5	4/14
	outlook=Overcast⇒Yes	0/4	
	outlook=Rain⇒Yes	2/5	
Temp	Temp=Hot⇒No	2/4	5/14
	Temp=Mild⇒Yes	2/6	
	Temp=Cool⇒Yes	1/4	
Humidity	Humidity=High⇒No	3/7	4/14
	Humidity=Normal⇒Yes	1/7	
Windy	Windy=Weak⇒Yes	2/8	5/14
	Windy=Strong⇒No	3/6	

https://www.slideshare.net/totoyou/covering-rulesbased-algorithm

Sequential Covering Algorithm: Contoh

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*Ski?*
~~D1~~	~~Sunny~~	~~Hot~~	~~High~~	~~Weak~~	~~No~~
~~D2~~	~~Sunny~~	~~Hot~~	~~High~~	~~Strong~~	~~No~~
~~D3~~	~~Overcast~~	~~Hot~~	~~High~~	~~Weak~~	~~Yes~~
~~D4~~	~~Rain~~	~~Mild~~	~~High~~	~~Weak~~	~~Yes~~
~~D5~~	~~Rain~~	~~Cool~~	~~Normal~~	~~Weak~~	~~Yes~~
D6	Rain	Cool	Normal	Strong	No
~~D7~~	~~Overcast~~	~~Cool~~	~~Normal~~	~~Strong~~	~~Yes~~
~~D8~~	~~Sunny~~	~~Mild~~	~~High~~	~~Weak~~	~~No~~
D9	Sunny	Cool	Normal	Weak	Yes
~~D10~~	~~Rain~~	~~Mild~~	~~Normal~~	~~Weak~~	~~Yes~~
D11	Sunny	Mild	Normal	Strong	Yes
~~D12~~	~~Overcast~~	~~Mild~~	~~High~~	~~Strong~~	~~Yes~~
~~D13~~	~~Overcast~~	~~Hot~~	~~Normal~~	~~Weak~~	~~Yes~~
D14	Rain	Mild	High	Strong	No

Relative Frequency:

$$\frac{n_c}{n}$$

n = # examples the rule matches

n_c = # examples the rule matches and classifies correctly

Threshold=0.5
LearnedRules={}

Rule R1: { outlook=Sunny⇒No;
outlook=overcast⇒Yes;
outlook=Rain⇒Yes}
RelativeFreq(R1,Examples)=10/14 > threshold

LearnedRules={R1}
Examples= {D6,D9,D11,D14}

Sequential Covering Algorithm: Contoh (lanj)

Day	Outlook	Temperature	Humidity	Wind	Ski?
D6	Rain	Cool	Normal	Strong	No
D9	Sunny	Cool	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D14	Rain	Mild	High	Strong	No

Rule R2: { outlook=Sunny⇒Yes;
outlook=Rain⇒No}
RelativeFreq(R2,Examples)=4/4 > threshold

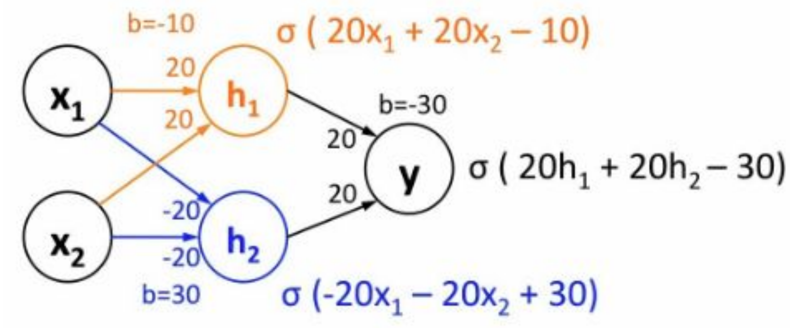
LearnedRules={R1,R2}
Examples= {}

Sorted LearnedRules: {R2,R1}

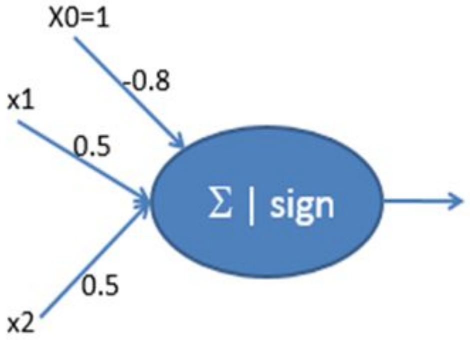
Attribute	Rules	Error	Total
Outlook	outlook=Sunny⇒Yes	0/2	0/4
	outlook=Rain⇒No	0/2	
Temp	Temp=Mild⇒Yes	1/2	2/4
	Temp=Cool⇒Yes	1/2	
Humidity	Humidity=High⇒No	0/1	1/4
	Humidity=Normal⇒Yes	1/3	
Windy	Windy=Weak⇒Yes	0/1	1/4
	Windy=Strong⇒No	1/3	

Klasifikasi dengan Artificial Neural Network (FFNN)

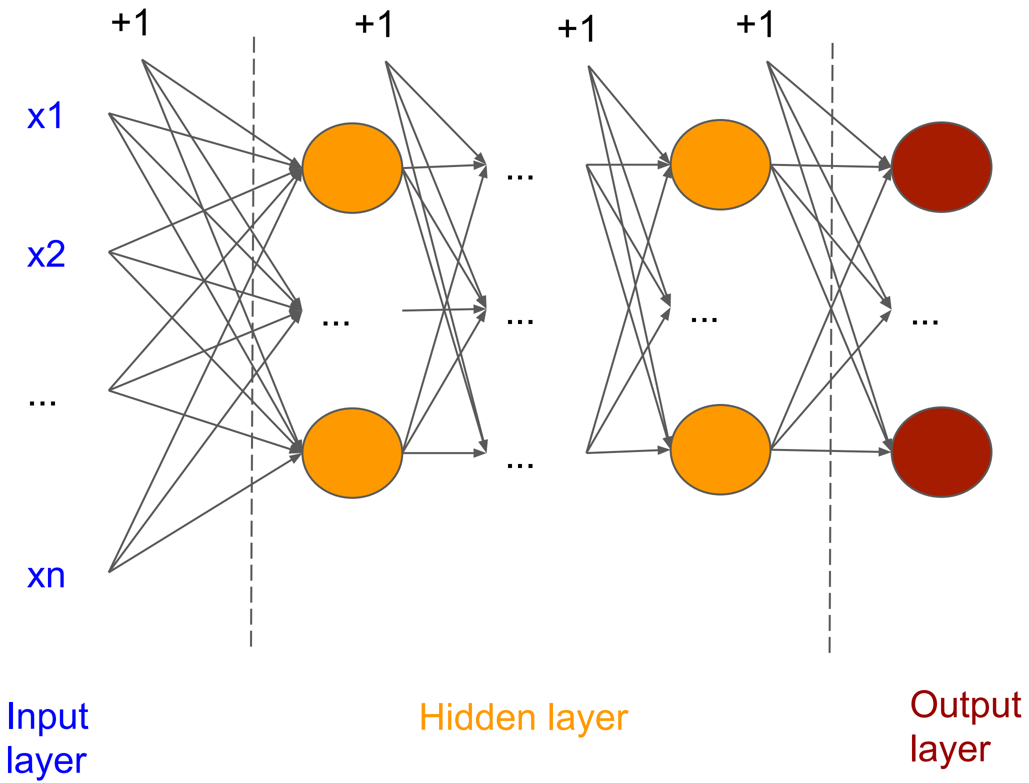
x1	x2	f
0	0	0
0	1	1
1	0	1
1	1	0



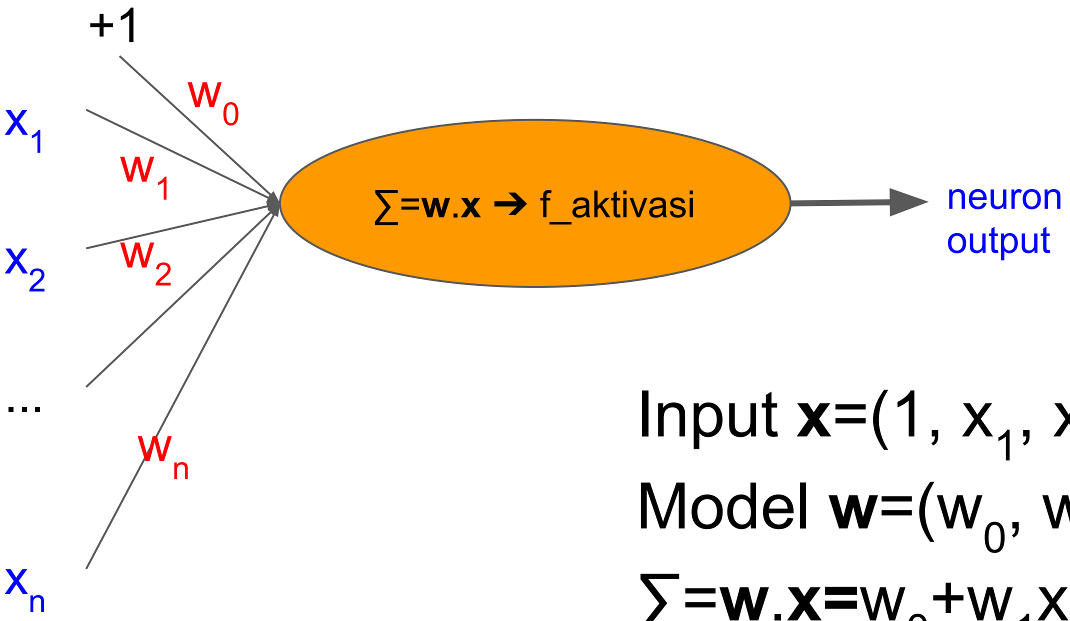
x1	x2	f
1	1	1
1	-1	-1
-1	1	-1
-1	-1	-1



Artificial Neural Network (Jaringan Syaraf Tiruan)



Artificial Neuron



Input $\mathbf{x}=(1, x_1, x_2, \ldots, x_n)$

Model $\mathbf{w}=(w_0, w_1, w_2, \ldots, w_n)$

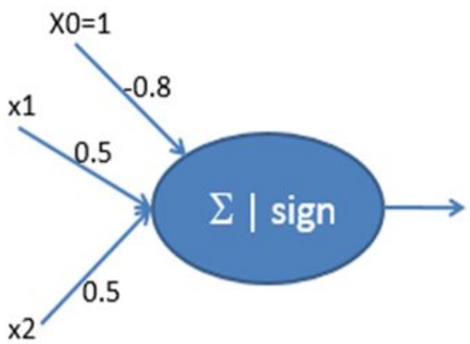
$\sum=\mathbf{w}.\mathbf{x}=w_0+w_1x_1+..+w_nx_n$

Output = $f\ (\sum)$

Input: real-valued vector

x_0=1 (bias)

Perceptron AND



x1	x2	∑	sign(∑)	f
1	1	-0.8*1+0.5*1+0.5*1=0.2	+1	1
1	-1	-0.8*1+0.5*1+0.5*-1=-0.8	-1	-1
-1	1	-0.8*1+0.5*-1+0.5*1=-0.8	-1	-1
-1	-1	-0.8*1+0.5*-1+0.5*-1=-1.8	-1	-1

∑= -0.8*x0+0.5*x1+0.5*x2
f = sign(∑)

sign (net): +1 jika net > 0, dan -1 sebaliknya

Fungsi Aktivasi

<u>Linear / identity</u>

f (net)=net

f’(net)=1

<u>Relu (rectified linear unit)</u>

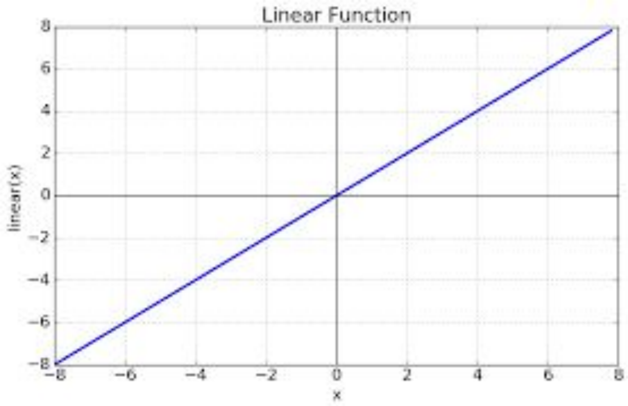
f (net)=max(0,net)

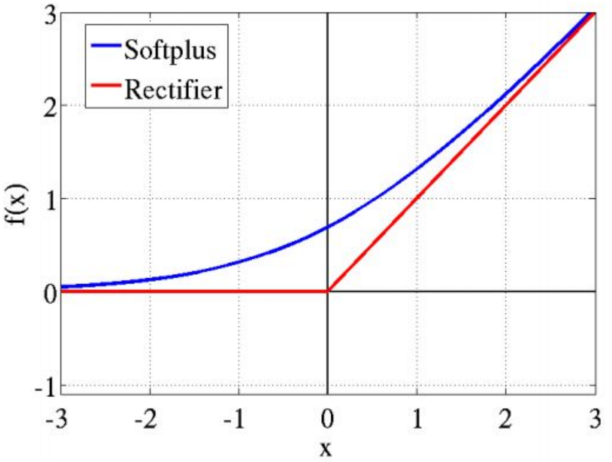
f’(net)=max(0,1)

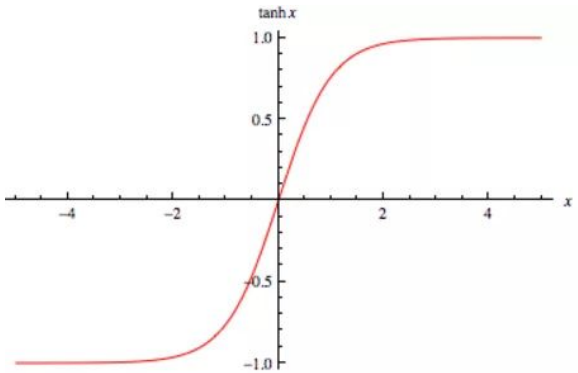
<u>tanh</u>

f (net)=(e^{net}-e^{-net})/(e^{net}+e^{-net})

f’(net)=1-(f (net))^2





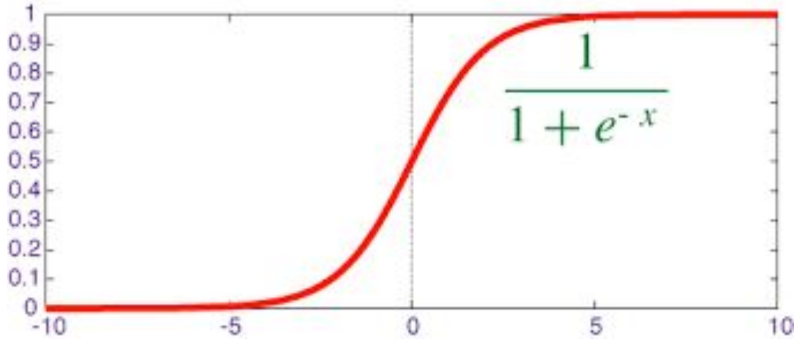


Fungsi Aktivasi (lanjutan)

Sigmoid / logistik /soft step

f (net)=1/(1+e^{-net})

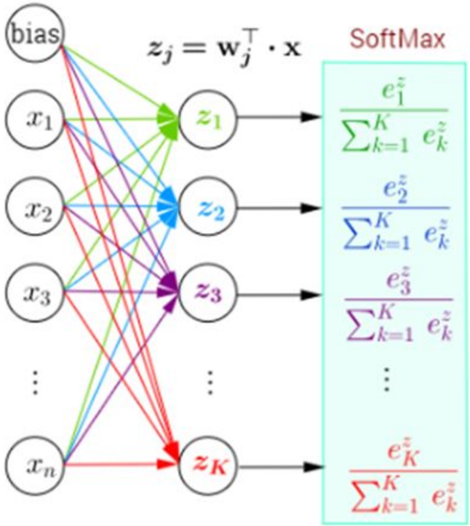
f'(net)=f (net)(1-f (net))



Softmax

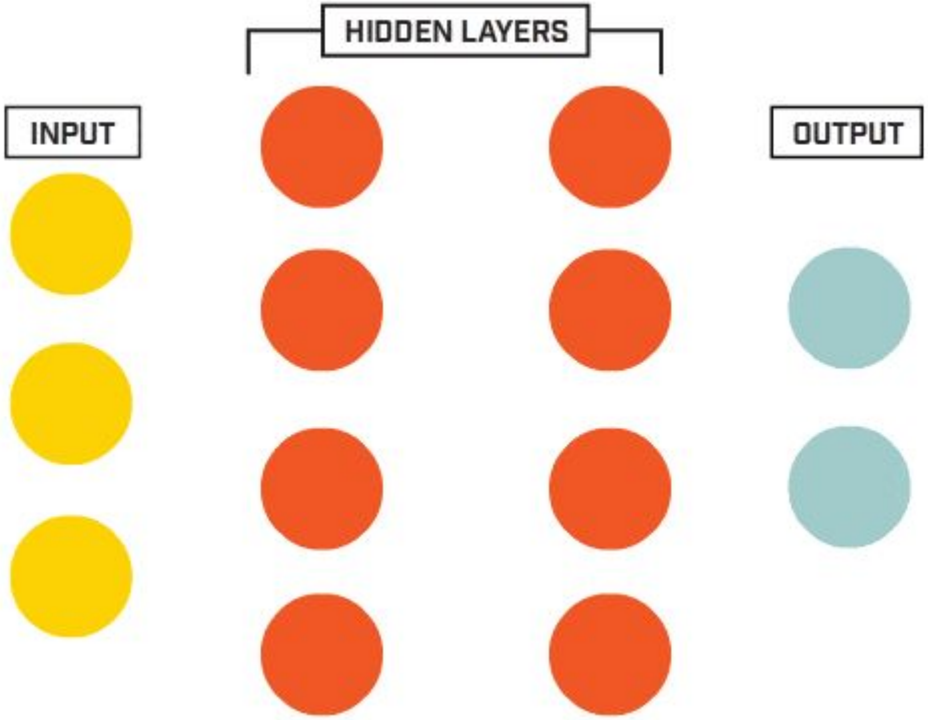
f (net_i)=e^{net_i}/∑e^{net_j}

f'(net_i)=f (net_i)(1-f (net_i))



https://isaacchanghau.github.io/post/activation_functions/

Feed Forward ANN

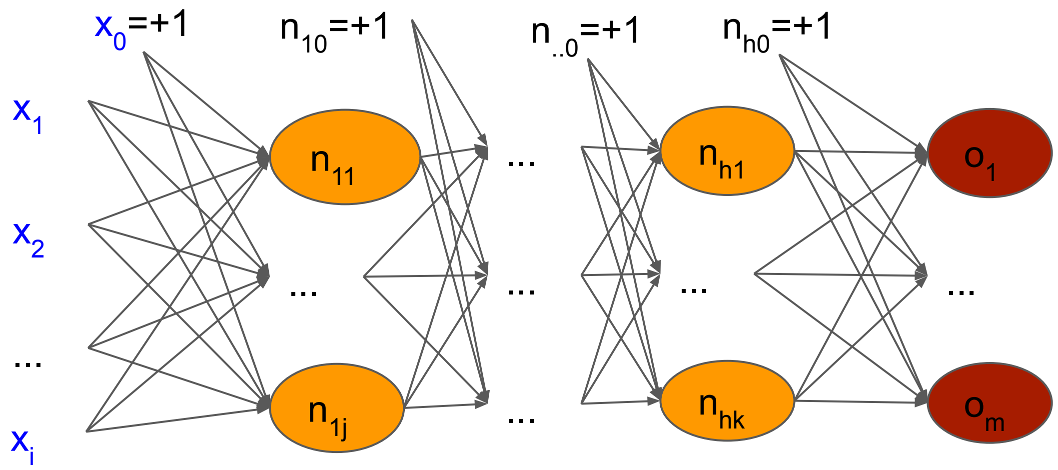


Struktur jaringan tanpa backward link

Informasi mengalir maju
input → hidden layers → output

https://kookycoder.com/blog/deep-learning-and-its-functioning/

Feed Forward ANN: Topologi

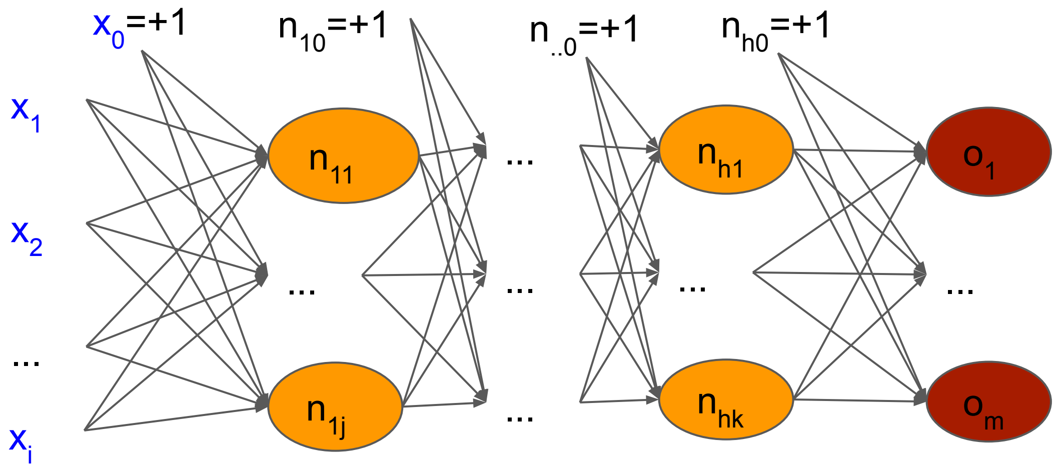


Input layer
berukuran i fitur

H hidden layer
Hidden layer 1: j neuron
Hidden layer h: k neuron

Output layer
berukuran m neuron

Feed Forward ANN: Representasi Model (Bobot)

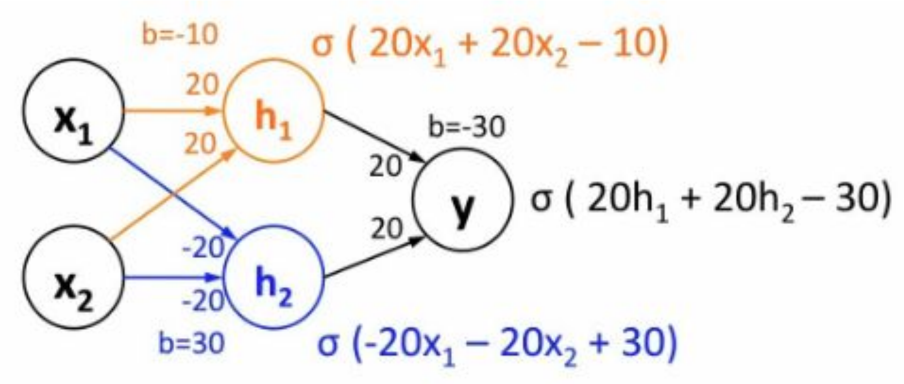


	n_{11}	n_{12}	...	n_{1j}
x_0=+1	w_{0_11}	w_{0_12}	...	w_{0_1j}
x_1	w_{1_11}	w_{1_12}	...	w_{1_1j}
x_2	w_{2_11}	w_{2_12}	...	w_{2_1j}
...	...	...	...	...
x_i	w_{i_11}	w_{i_12}	...	w_{i_1j}

	n_{21}	n_{22}	...
n_{10}=+1	w_{10_21}	w_{10_22}	...
n_{11}	w_{11_21}	w_{11_22}	...
n_{12}	w_{12_21}	w_{12_22}	...
...	...	...	...
n_{1j}	w_{1j_21}	w_{1j_22}	...

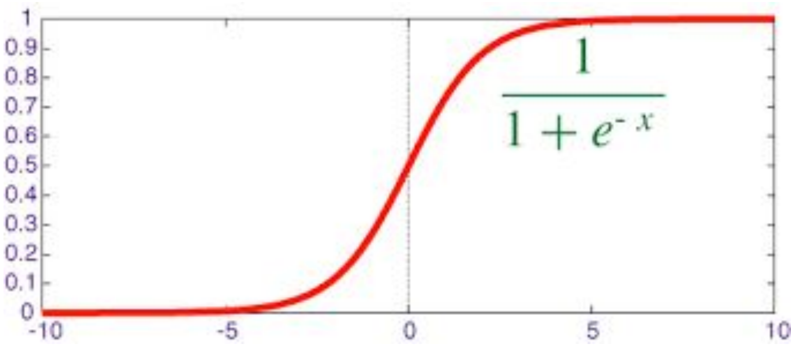
	o_1	...	o_m
n_{h0}=+1	w_{h0_1}	...	w_{h0_m}
n_{h1}	w_{h1_1}	...	w_{h1_m}
n_{h2}	w_{h2_1}	...	w_{h2_m}
...	...	...	...
n_{hk}	w_{hk_1}	...	w_{hk_m}

Feed Forward ANN: Contoh Klasifikasi



<u>Sigmoid / logistik /soft step</u>

$f\ (net) = 1/(1+e^{-net})$



x1	x2	f
0	0	0
0	1	1
1	0	1
1	1	0

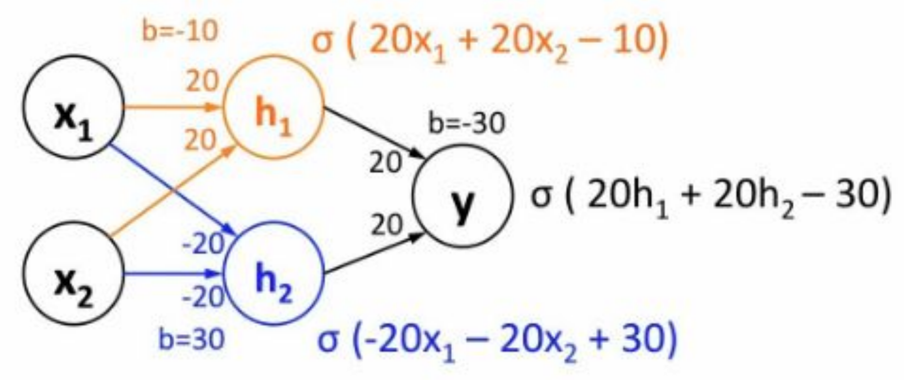
x1=1; x2=0:

$h1=\sigma(20x1+20x2-10)=\sigma(20*1+20*0-10)=\sigma(10)\approx 1$

$h2=\sigma(-20x1-20x2+30)=\sigma(-20*1-20*0+30)=\sigma(10)\approx 1$

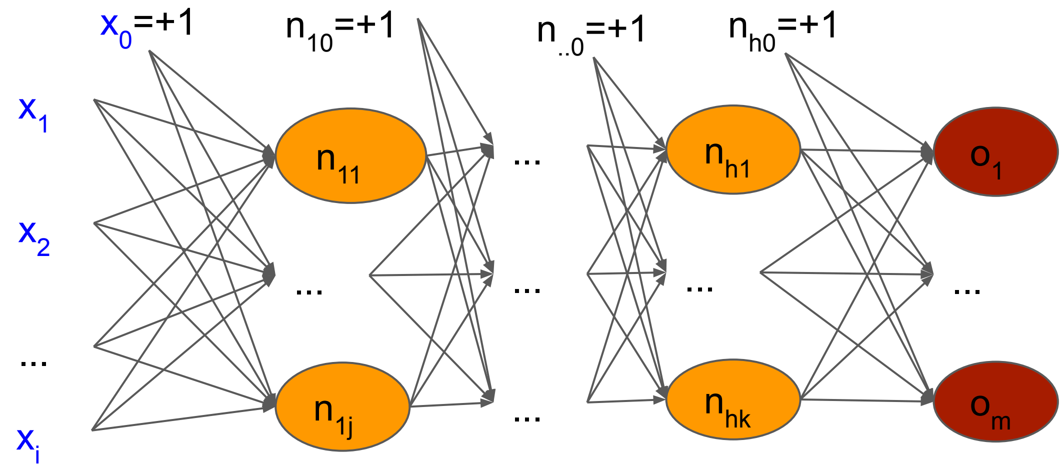
$y=\sigma(20h1+20h2-30)=\sigma(20*1+20*1-30)=\sigma(10)\approx \mathbf{1}$

Feed Forward ANN: Contoh Klasifikasi



x0	x1	x2	f	∑h1	h1	∑h2	h2	∑y	y
1	0	0	0	-10	0.00	30	1.00	-10.00	0.00
1	0	1	1	10	1.00	10	1.00	10.00	1.00
1	1	0	1	10	1.00	10	1.00	10.00	1.00
1	1	1	0	30	1.00	-10	0.00	-10.00	0.00

Feed Forward ANN: Perhitungan Output



Input $\mathbf{x}=(1, x_1, x_2, \ldots, x_i)$

$n_{11} = f\ ((w_{0\_11}, \ldots, w_{i\_11}).(1, x_1, \ldots, x_i))$

…

$n_{1j} = f\ ((w_{0\_1j}, \ldots, w_{i\_1j}).(1, x_1, \ldots, x_i))$

$n_{21} = f\ ((w_{10\_21}, \ldots, w_{1j\_21}).(1, n_{11}, \ldots, n_{1j}))$

…

$o_1 = f\ ((w_{h0\_1}, \ldots, w_{hk\_1}).(1, n_{h1}, \ldots, n_{hk}))$

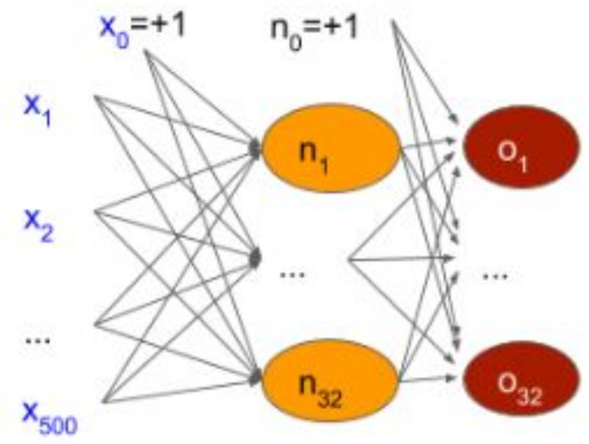
…

$o_m = f\ ((w_{h0\_m}, \ldots, w_{hk\_m}).(1, n_{h1}, \ldots, n_{hk}))$

	n_{11}	n_{12}	...	n_{1j}
x_0=+1	w_{0_11}	w_{0_12}	...	w_{0_1j}
x_1	w_{1_11}	w_{1_12}	...	w_{1_1j}
...	...	...	...	...
x_i	w_{i_11}	w_{i_12}	...	w_{i_1j}

	o_1	...	o_m
n_{h0}=+1	w_{h0_1}	...	w_{h0_m}
n_{h1}	w_{h1_1}	...	w_{h1_m}
...	...	...	...
n_{hk}	w_{hk_1}	...	w_{hk_m}

Feed Forward ANN pada Keras




```
from keras import models
from keras import layers

#definisikan model sebagai urutan layer
model = models.Sequential()
#model menerima input dengan jumlah fitur 500 dan output 32
#jumlah instance tidak perlu dituliskan dalam input_shape
model.add(layers.Dense(32,input_shape=(500,)))
#input layer ini otomatis 32 (output layer sebelumnya)
model.add(layers.Dense(32))
```

Tugas 1

- Pilihlah dataset dari https://archive.ics.uci.edu/ml/datasets.php
- Tools: Jupyter Notebook dengan python. Gunakan scikit-learn untuk konstruksi model klasifikasi (training data)
- Skenario:
  - split dataset menjadi 90:10 (train:test)
  - Lakukanlah holdout validation dengan split training data menjadi 90:10 (train:validasi)
    - Gunakanlah validation data untuk memilih model klasifikasi terbaik dari berbagai algoritma pembelajaran dengan variasi nilai parameter berbeda.
  - Lakukanlah 5-fold cross validation untuk training data.
    - Pilih model klasifikasi terbaik dari berbagai algoritma pembelajaran dengan variasi nilai parameter berbeda.
  - Gunakanlah test data untuk mendapatkan kinerja dari setiap skenario.
- Lakukanlah analisis di proses validasi dan testing