Short REVIEW for FINAL

Professor Anita Wasilewska
Computer Science Department
Stony Brook University
Part 1: PREDICATE LOGIC CONCEPTUALIZATION

- Translations from Natural Language
- BE CAREFUL!
- YOU MUST ALWAYS DO DIRECT TRANSLATION
- Never translate some logically EQUIVALENT FORM like in this case (via de Morgan Laws)
- “All houses are not red”
• Translations from Natural Language

• Translate: “All houses are not red”

• 1. Domain: $X \neq \emptyset$
• 2. Predicates: $A(x) – x$ is a house   $B(x) – x$ is red
• 3. Functions: (none)
• 4. Connectives: $\neg$ - “not”
• 5. Quantifiers: $\forall_{A(x)} – “All$ houses” (restricted)
• 6. RESTRICTED FORMULA: $\forall_{A(x)} \neg B(x)$
• 7. LOGIC FORMULA: $\forall_x (A(x) \implies \neg B(x))$
PART 1: PREDICATE LOGIC CONCEPTUALIZATION

- Translations from Natural Language
- Translate: “No house is red”

1. Domain: $X \neq \emptyset$
2. Predicates: $A(x) \rightarrow x$ is a House  $B(x) \rightarrow x$ is red
3. Functions: (none)
4. Connectives: $\neg$ - “not”
5. Quantifiers: $\exists_{A(x)} \rightarrow$ “some houses” (restricted)
6. RESTRICTED FORMULA: $\neg \exists_{A(x)} B(x)$
7. LOGIC FORMULA: $\neg \exists x (A(x) \land B(x))$
Part 2: Propositional Resolution
GOAL: Use Resolution to prove/ disapprove $\vdash A$

PROCEDURE

Step 1: Write $\neg A$ and transform $\neg A$ into a set of clauses $\text{CL}_{\neg A}$ using Transformation rules

Step 2: Consider $\text{CL}_{\neg A}$ and look at if you can get a deduction of $\{\}$ from $\text{CL}_{\neg A}$

ANSWER

1. $\text{CL}_{\neg A} \vdash \{}$ – Yes, $\vdash A$
2. $\text{CL}_{\neg A} \not\vdash \{}$ (i.e. you never get $\{}$) – No, not $\vdash A$
Rules of transformation of a formula $A$ into a logically equivalent set of clauses $\text{CL}_A$

- **Rule (U):** $(A \cup B) + \text{Information}$

What “Information” mean?

Example: $a, b, (a \cup \neg( a \Rightarrow b)), \neg c$

$$a, b, a, \neg( a \Rightarrow b), \neg c$$

$a, b, \neg c$ is Information

Rule (U): $I, (A \cup B), J$

$I, A, B, J$

$I, J$ --- Information around
Implication Rule (\(\Rightarrow\))

- I, (A\(\Rightarrow\)B), J

\[ (A\Rightarrow B) \]

I, \(\neg A\), B, J

\[ \neg A\), B \]

Example:

\[ a, (a \cup b), (a \Rightarrow \neg a), (a \land b), c \]

\[ (=\Rightarrow) \]

\[ a, (a \cup b), \neg a, \neg a, (a \land b), c \]

\[ (=\cup) \]

\[ a, a, b, \neg a, \neg a, (a \land b), c \]

next step?

we need \((\land)\) Rule!
Conjunction Rule ($\land$)

I, (A $\land$ B), J

(A $\land$ B)

I, A, J   I, B, J

Example:

a, a, b, ¬a, ¬a, (a $\land$ b), c

(a $\land$ b), c

a, a, b, ¬a, ¬a, a, c

(STOP) when get only literals

Form clauses out of the leaves
Set of Clauses

Procedure: Leaves – to – Clauses
1. make SETS out of each leaf; each leaf becomes a clause $C$
2. make a set of clauses $CL$ as a set of all clauses $C$ obtained in 1.
   Leaf 1: $\{a, a, b, \neg a, \neg a, a, c\} = \{a, b, \neg a, c\}$
   Leaf 2: $\{a, a, b, \neg a, \neg a, b, c\} = \{a, b, \neg a, c\}$
• Observe that we end-up with only one set of clauses
• $CL = \{\text{Leaf 1, Leaf 2}\} = \{ \{a, b, \neg a, c\}\}$
Negation of Implication Rule ($\neg =>$)

$$\neg (A => B), \neg (A => B)$$

Example:

$$a, b, a, \neg (a => b), \neg c$$

Stop – when only literals:

Form clauses out of: $a, b, a, a, \neg c$ and $a, b, a, \neg b, \neg c$
Clauses

• Leaf1: $a, b, a, a, \neg c$ makes clause $\{a, b, \neg c\}$
• Leaf 2: $a, b, a, \neg b, \neg c$ makes clause $\{a, b, \neg b, c\}$

• $CL = \{\{a, b, \neg c\}, \{a, b, \neg b, c\}\}$

• $CL$ is set of clauses corresponding to
  $a, b, a, \neg (a \Rightarrow b), \neg c$
Negation of Conjunction Rule ($\neg \land$)

$I, \neg(A \land B), J \quad \neg(A \land B)$

$(\neg \land)$

$I, \neg A, \neg B, J \quad \neg A, \neg B$

Corresponds to DeMorgan Law

$\neg(A \land B) \equiv (\neg A \lor \neg B)$
Negation of Disjunction Rule ($\neg U$)

$I, \neg(A \lor B), J$

$\neg(A \lor B) \\ (\neg U)$

$I, \neg A, J \\ I, \neg B, J \\ \neg A \\ \neg B$

- Corresponds to DeMorgan Law:
  
  $\neg(A \lor B) \equiv (\neg A \land \neg B)$
Negation of Negation Rule ($\neg\neg$)

$I, \neg\neg (A), J \quad \neg\neg (A)$

$\neg\neg (A)$

$I, A, J \quad A$

Corresponds to

$\neg\neg (A) \equiv A$

Transformation Rules:

$(\land), (\lor), (\Rightarrow), (\neg \land), (\neg \lor), (\neg \Rightarrow)$
Transformation Rules Shorthand Form

\[(A \cup B) \quad (U)\]
\[A, \ B\]

\[(A \land B) \quad (\land)\]
\[A \quad B\]

\[(A \Rightarrow B) \quad (\Rightarrow)\]
\[\neg A, \ B\]

\[\neg \neg A \quad (\neg \neg)\]
\[A\]

\[-(A \cup B) \quad (-U)\]
\[-A \quad -B\]

\[-(A \land B) \quad (-\land)\]
\[-A, \ -B\]

\[-(A \Rightarrow B) \quad (-\Rightarrow)\]
\[A \quad -B\]

+ Keep all information

End when all leaves are literals
ARGUMENTS (rules of inference)

• From (premises) \( A_1, \ldots, A_n \) we conclude B

\[
\begin{align*}
A_1, \ldots, A_n \\
\text{B}
\end{align*}
\]

Definition:

Argument \( A_1, \ldots, A_n \) is VALID iff

\[
| = ((A_1 \land \ldots \land A_n) \Rightarrow B)
\]
ARGUMENTS

• Otherwise
  Argument is NOT VALID

Valid Arguments ≡ Tautologically Valid
$A_1, \ldots, A_n, C$
are formulas of Propositional or Predicate Language
Validity of Arguments

Remember:  \( |= A \iff |-A \)

Tautology (always true), Contradiction (always false)

This means that if we want to decide \( |= A \) we decide \( |-A \)
and use Resolution for that

**STEPS**

**Step 1:** Negate \( A \); i.e. take \( |-A \) and find the set of clauses corresponding to \( |-A \) i.e. find \( CL\{ -A \} \)

**Step 2:** Use Completeness of Resolution

\( |= A \iff CL\{ -A \} \vdash R \{ \} \) i.e.

1. Look for a deduction of \( \{ \} \)
2. if YES – we have \( |= A \)
3. If there is no deduction of \( \{ \} \) we have: \( |= A \)
Exercise

• **Prove** By Propositional Resolution
• \(|= (\neg(a=>b) => (a \land \neg b))\)

**Remember:** \(|= A \iff =| \neg A + \text{use Resolution}  

**Steps**

**Step 1:** Find set of clauses corresponding to \(\neg A\)
  i.e. \(\text{CL}_{\neg A}\)

**Step 2:** Find deduction of \(\{\}\) from  . \(\text{CL}_{\neg A}\)
  i.e. show that \(\text{CL}_{\neg A} \vdash_R \{\}\)

DO IT!
Exercise Solution

• **Step 1:** Negate $A$ and find the set of clauses for $\neg A$ i.e. $\text{CL}_{\neg A}$

  $\neg (\neg (a \Rightarrow b) \Rightarrow (a \land \neg b))$

  $\neg (a \Rightarrow b)$  $\quad$  $\neg (a \land \neg b)$

  $a$  $\neg b$  $\neg a, \neg \neg b$

  $\{a\}$  $\{\neg b\}$  $\{\neg a, b\}$

  $\text{CL}_{\neg A} = \{\{a\}, \{\neg b\}, \{\neg a, b\}\}$

  Step 2: Check if $\text{CL}_{\neg A} \vdash R \{\}$ — YES!

Remark: $|\vdash A$ iff there is no deduction of $\{\}$ from $\text{CL}_{\neg A}$
• Use resolution to show that from $A_1, \ldots, A_n$ we can deduce $B$

"We can" deduce $B$ from $A_1, \ldots, A_n$ means validity of argument $A_1, \ldots, A_n \vdash B$

iff by definition

$| = (A_1 \land \ldots \land A_n \rightarrow B)$

We have to use Resolution to prove that this is a Tautology
Arguments

\(|= (A_1 \land \ldots \land A_n \Rightarrow B) \iff \neg (A_1 \land \ldots \land A_n \Rightarrow B) \iff \neg (A_1 \land \ldots \land A_n \land \neg B)\)

- **Step 1:** we transform \((A_1 \land \ldots \land A_n \land \neg B)\) to clauses

- Take \(A_1, \ldots, A_n\) and find \(CL_{A_1}, \ldots, CL_{A_n}\) and also find \(CL_{\neg B}\) and form

\(CL_{A_1} \cup \ldots \cup CL_{A_n} \cup CL_{\neg B} = CL\)

Step 2: examine whether \(CL \vdash_R \{}\)
Remember

• Argument $A_1, \ldots, A_n$ is valid iff

$$\text{B} \quad \text{CL}_{A_1} \cup \ldots \cup \text{CL}_{A_n} \cup \text{CL}_{\neg B} \vdash_R \{\}$$

\[ \downarrow \]

Argument is not valid iff never $\text{CL}_{A_1} \cup \ldots \cup \text{CL}_{A_n} \cup \text{CL}_{\neg B} \vdash_R \{\}$

We have some Resolution Strategies that allow us to cut down number of cases to consider
Part 3: Classification Learning Process

- Classification process operate in three stages:
  
  **Stage 1:** build the basic patterns structure - training
  
  **Stage 2:** optimize parameter settings; can use (N:N) re-substitution - parameter tuning
  
  **Stage 3:** use test data to compute predictive accuracy/error rate
Classifier, Model Terminology

- Books use the words "classifier" and "model" interchangeably
- Sometimes "classifier" means Stage 1 basic classifier model (rules, patterns) ready for **testing**
- Sometimes "classifiers" means classifiers models (rules, patterns) obtained by training - testing methods (like k-fold cross validation, repeated holdout, etc..). i.e. are the results of Stages 1-3
Classifier, Model Terminilogy

• In some cases the term “learned models”
• or “base classifiers” are used for results of
• Stages 1-3

• It happens when the method is presented how to combine them in a way that would the best to return a class prediction for unknown records, i.e. to build the final

• CLASSIFIER
TESTING

Define a holdout procedure

- Holdout procedure is a method of splitting original data into training and test data sets
Describe shortly the two main methods of **predictive accuracy** evaluations

(1) **k-fold cross-validation** \((N - N/k ; N/k)\)

(2) **Leave-one-out** \((N - 1 ; 1)\)
(1) **k-fold cross-validation** \((N - N/k ; N/k)\)

First step:

**split data** into **k disjoint** subsets

\(D1, \ldots , Dk,\)

of equal size, called **folds**

Second step:

**use each** subset in turn for **testing**, the remainder for **training**

**Training and testing** is performed **k times**
Testing

• (2) **Leave-one-out (N-1;1)**

Leave-one-out is a particular form of cross-validation

• We set number of **folds** to number of training instances, i.e. we put \( k = N \) for \( N \) instances

• We **repeat** the **training – testing** cycle \( N \) times
Correctly and Not Correctly Classified

- A test data record is correctly classified if and only if the following conditions hold:
  1. we can classify the record, i.e. there is a pattern or a rule such that its LEFT side matches the record,
  2. classification determined by the pattern or the rule is correct, i.e. the RIGHT side of the rule matches the value of the record’s class attribute

OTHERWISE
- the record is not correctly classified

- Words used:
  - not correctly = incorrectly = misclassified
  - Validation data = Test data
Re-substitution Error Rate

- **Re-substitution error rate** is obtained from **training data**
- **Training Data Error**: uncertainty of the rules
- The error rate is **not always 0%**, but usually (and hopefully) very low!
- **Re-substitution error rate** indicates only how **good (bad)** are our **results** (rules, patterns, NN) on the **TRAINING data**
- It expresses some **knowledge** about the algorithm used
Re-substitution Error Rate

• Re-substitution error rate is usually used as the performance measure:

  The training error rate reflects imprecision of the training results.
  
  The lower training error rate the better.
  
  In the case of rules it is called rules accuracy.
Predictive Accuracy

**Predictive accuracy** reflects how **good** are the **training results** with respect to the **test data**.

The higher predictive accuracy **the better**

**(N:N)** re-substitution **does not** compute predictive accuracy

- Re-substitution error rate = training data error rate
Validation Data

- Proper **classification process** uses three sets of data:
  - training data, validation data and test data
  - validation data is **used** for parameter tuning
  - validation data is **not** a test data
  - validation data can be the **training data**, or a subset of training data
  - The **test data** can not be used for parameter tuning!
Classifier, Model Terminology

• When a book talks about **comparison of classifiers**, “classifier” means comparison of classifiers models (rules, patterns) obtained by **train-test methods** i.e. means comparison results of Stages 1-3

• These **comparison methods** or other methods are called “**model selection**”

• Their goal is to **choose** the best one to be

• **THE CLASSIFIER**-

• the final product that would the best **classify** unknown records