Laurea Specialistica in Informatica
a.a. 2007-2008

Interazione Uomo - Macchina II:

Laboratorio di Interfacce Intelligenti

Fiorella de Rosis

Esercitazione 2

Formalizzazione di un modello utente mediante RCP

Tutor esercitazione: Irene Mazzotta

Prerequisiti

Concetti di base su Reti Causali Probabilistiche, Ragionamento incerto e Modelli di Utente

Materiale

Dispense del corso, Unità 5 e 6 (reperibili sul sito web)

Obiettivo

- Formalizzare un modello di utente con incertezza mediante la costruzione di una rete causale probabilistica (RPC).
- Simulare ragionamento 'diagnostico' e 'prognostico' mendiante la propagazione delle evidenze (Algoritmo di Spiegelhalter).
- Rappresentazione di un modello dinamico mediante una RPC dinamica
- Esperienza con Hugin Lite

Reti Causali Probabilistiche (RCP) (o Belief Network, oBayesian Network)

- *Bayesian network* (*BN*) è un formalismo per modellare un dominio contenente varie forme di incertezza.
- A BN è un grafo orientato, aciclico, i cui nodi rappresentano variabili a più valori e gli archi rappresentano la *relazione causale* fra i nodi che collegano.
- La forza di queste relazioni è misurata in termini di *probabilità condizionate*.
- Ad ogni nodo radice (che non ha genitori), è associata una *tabella di probabilità marginale*. A tutti gli altri nodi è associata una *tabella di probabilità condizionata*

Tool per BN – info utili

Hugin:
sw commerciale per Belief Network.
Fornisce API per C, C++, Java.

eBayes/javaBayes
sw open source completamente scritto in java

Netica
sw commerciale con API per C, Java e VB.

Analytica
sw commerciale per la creazione, analisi e comunicazione di decisioni con BN e Influence Diagram.

Genie, MSBN, ecc…

Hugin components

Hugin Development Environment ha tre componenti:

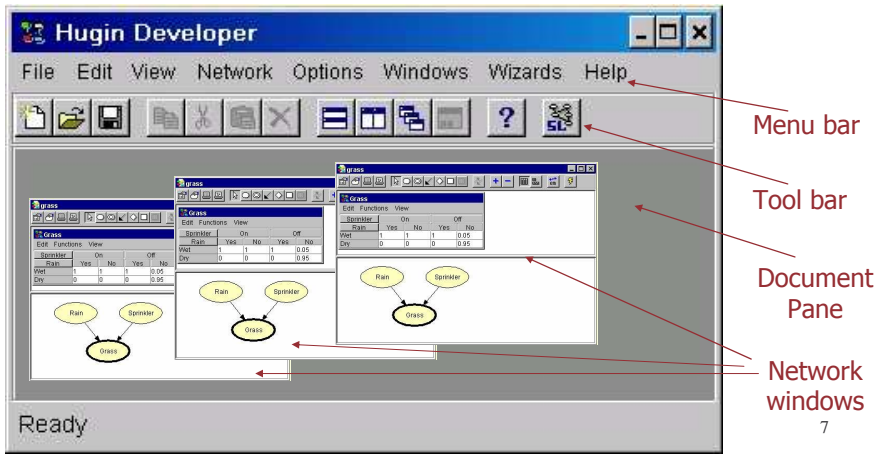
- ***Hugin Decision Engine*** (HDE): è il motore inferenziale di Hugin; esegue il ragionamento sulla KB rappresentata mediante Bayesian Network; una parte importante dell'HDE è il *compiler* che trasforma le reti in strutture funzionali (junction trees), rendendo possibile le inferenze (reasoning) in the network.
- ***Hugin Graphical User Interface*** e ***Hugin Application Program Interfaces*** (API) rappresentano i due possibili accessi all'HDE.

Hugin Graphical User Interface

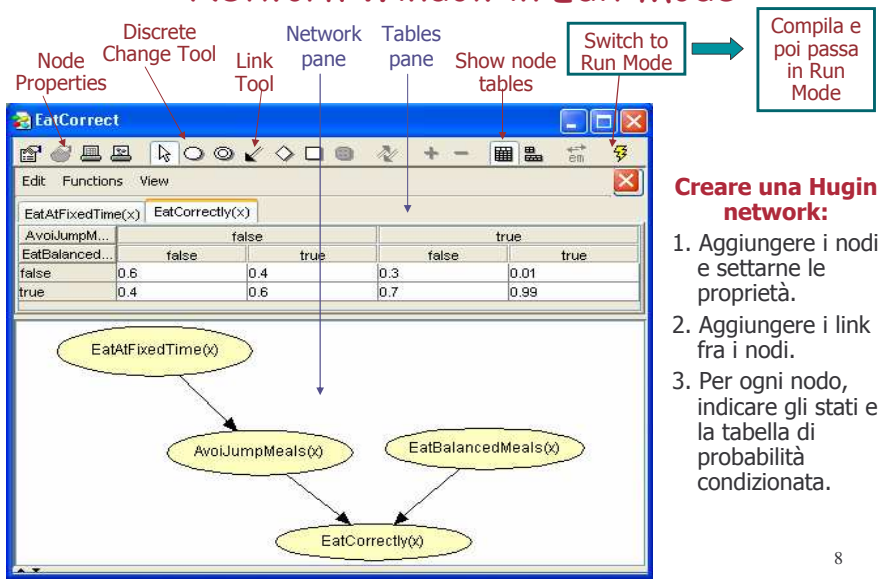
- *Hugin Graphical User Interface* è usato per creare, compilare ed eseguire (inserendo e propagando evidenze) modelli di network.
- Consiste di due modalità operative: *Edit Mode* and *Run Mode*.
- La modalità *edit* è usata per creare i nodi e link fra questi, gli stati dei singoli nodi, le tabelle di probabilità condizionata e marginale. Tutte queste operazioni vengono eseguite mediante *a window-, menu- and mouse driven interface*.
- Nella modalità *Run* l'utente può inserire evidenze sui nodi osservabili attraverso la selezione degli stati interessati. Hugin Decision Engine provvederà a propagare le informazioni inserite e a rivedere le probabilità dei nodi coinvolti.

Hugin Graphical User Interface: Main Window

Contiene le *Network Windows* ognuna delle quali visualizza una Hugin network e può operare in *Edit Mode* o in *Run Mode*.



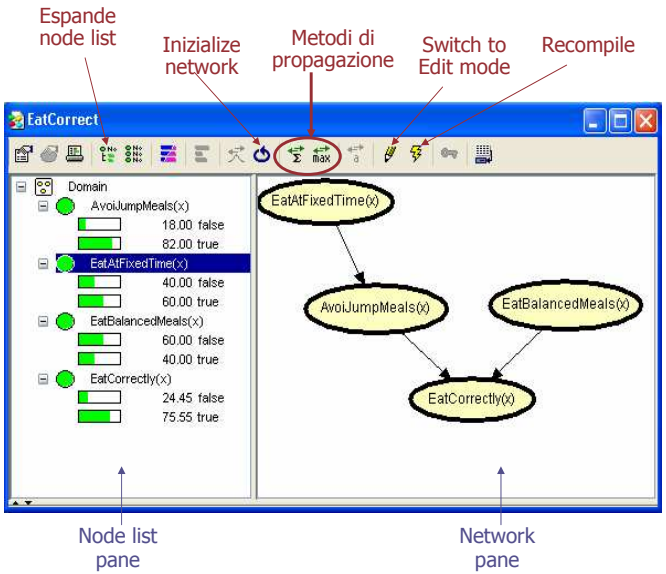
Costruire una Hugin Network: Network Window in Edit Mode



Creare una Hugin network:

1. Aggiungere i nodi e settarne le proprietà.
2. Aggiungere i link fra i nodi.
3. Per ogni nodo, indicare gli stati e la tabella di probabilità condizionata.

Usare una Hugin Network: Network Window in Run Mode



Usare una Hugin network:

1. Introdurre l'evidenza nella rete (node list pane)
2. Propagare l'evidenza (Algortmo di Spiegelhalter)
3. Osservare l'effetto sulla probabilità delle altre variabili (*Prob a Posteriori*).

Uno stralcio dal file ".hlg"

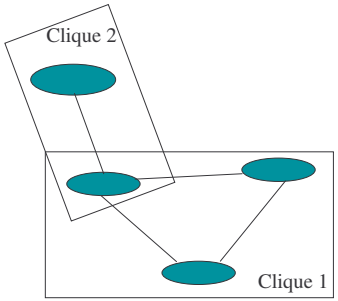
Moral links:
Marrying AvoiJumpMeals and EatBalancedMeals

Triangulating prime component with 2 members:
AvoiJumpMeals, EatAtFixedTime
This component is a clique of cost 4

Triangulating prime component with 3 members:
EatCorrectly, EatBalancedMeals, AvoiJumpMeals
This component is a clique of cost 8

Total cost of triangulation for all prime components is 12

Cliques:
Clique 2, 2 members, table size = 4:
AvoiJumpMeals, EatAtFixedTime
Clique 1, 3 members, table size = 8:
AvoiJumpMeals, EatCorrectly, EatBalancedMeals
Total clique table size: 12



The junction forest:
Creating junction tree with clique 2 as root ...
Cliques 1 and 2 linked, separator has 1 member and table size = 2:
AvoiJumpMeals

Hugin APIs

- The *Hugin APIs* (Application Program Interfaces): per costruire applicazioni knowledge-based, che sfruttano la potenza dell'HDE quale motore inferenziale.
- Forniscono i metodi per inserire il motore inferenziale di Hugin all'interno di un'applicazione che abbia come base di conoscenza dei BNs
- Hugin APIs sono disponibili come C, C++, e Java libraries e come ActiveX server.
- Una breve lista delle più importanti funzioni fornite dall'HDE attraverso le APIs è disponibile alla pagina Feature Lists del web site di Hugin. Inoltre, Hugin API reference manuals possono essere scaricate dal web site di Hugin.

Api JAVA di Hugin

- La libreria delle HUGIN API si trovano in due file:
  - hapi63.jar : l'interfaccia Java alla libreria sottostante in C. deve trovarsi nel **CLASSPATH**
  - libhapi63.dll : la libreria che deve trovarsi in una sottodirectory dell'applicazione o come argomento della VM con l'opzione **-Djava. library.path**
- La classe che dovra' gestire l'interazione con le API dovra' contenere l'import relativo
  - import COM.hugin.HAPI.*;

Api JAVA di Hugin: le classi base

- Domini, nodi ecc sono modellati come classi.
- I metodi della classe ***domain*** permettono di gestire una rete (creare un dominio vuoto/da file, compilarlo, salvarlo, propagare le evidenze, ecc.)
- I metodi della classe ***node*** permettono di gestire I nodi all'interno della rete (aggiungere, togliere, modificare attributi, settare le evidenze, attraversare il grafo, ecc.)

In appendice, un approfondimento dei princili metodi utilizzati per manipolare una o più Bayesian Network.

Api di Hugin: creare un dominio

- Per inserire i nodi e' necessario definire un dominio;
- Un dominio contiene il belief network che verra' elaborato. Questo puo' essere creato da codice o caricato da un file:
  - Es1: dominio1 = new Domain(); // crea un dominio vuoto
  - Es2: dominio2 = new Domain("nomefile.hkb") // carica un dominio da file
- La creazione di un dominio puo' essere inserita in un try – catch per verificare la corretta inizializzazione e individuare gli errori.

Api di Hugin: esempio

```
try {
dominio = new Domain("rete.hkb");
   }

   catch (ExceptionHugin EH) {
     System.out.println(EH.getMessage());
     System.out.println("errore1");
     EH.printStackTrace(System.out);
     System.exit(1);
   }
```


Api di Hugin: I nodi

- Sia che il dominio sia vuoto o che lo si carichi da file e' necessario definire degli oggetti della classe **node** che conterrano i loro valori. I nodi possono essere di tipi diversi:
  - BooleanDCNode, (true-false)
  - IntervalDCNode, (0-1,1-2, ecc.)
  - LabelledDCNode, (etichettati)
  - NumberedDCNode (etichettati con un numero)

Es:...LabelledDCNode nodoEtichettatoN;
BooleanDCNode nodoBooleanoM; ...

Api di Hugin: leggere I nodi

- Agli oggetti così creati vanno associati I valori del dominio caricato dal file nel seguente modo:

```
try {…
nodoEtichettatoN =
   (LabelledDCNode)dominio.getNodeByName("EtichettaN");
nodoBooleanoM =
   (BooleanDCNode)dominio.getNodeByName("BooleanM");

      catch (ExceptionHugin ex) {
   System.out.println(ex.getMessage());
   System.out.println("errore2");
   ex.printStackTrace(System.out); System.exit(1);
      }
```


Api di Hugin: settare I nodi

- Se si aggiunge un nodo al dominio bisogna settare le informazioni del nodo.
  - Impostare il numero di stati:
    - setNumberOfStates (int newNumber)
  - Impostare l'etichetta di un nodo:
    - setStateLabel(int state, java.lang.String newLabel)
  - Definire I nodi genitori del nodo:
    - addParent (Node nomeNodoGenitore)

- Per eliminare un nodo si usa il metodo **delete()**

Api di Hugin: metodi utili (1)

- E' possibile accedere a tutte le informazioni su un nodo:
- NodeList getChildren() restituisce una lista dei nodi figli
- NetworkModel **getHome**() restituisce la classe o il dominio a cui appartiene il nodo
- Class **getHomeClass**() restituisce la classe che contiene il nodo
- Domain **getHomeDomain**() restituisce il dominio che contiene il nodo

Api di Hugin: metodi utili (2)

- NetworkModel.Kind **getKind**() restituisce il tipo di nodo (discreto, continuo, ecc)
- String **getLabel**() restituisce l'etichetta del nodo
- String **getName**() restituisce il nome del nodo
- NodeList **getParents**() restituisce una NodeList con i nodi genitori
- java.awt.geom.Point2D **getPosition**() Restituisce la posizione del nodo

Api di Hugin: metodi utili (3)

- Si puo' controllare lo stato di un nodo :
- boolean **evidenceIsEntered**()
  - Vero se e' stata imposta un evidenza sul nodo
- boolean **evidenceIsPropagated**()
  - vero se l'evidenza e' gia' stata propagata
- boolean **evidenceToPropagate**()
  - vero se l'evidenza imposta deve essere ancora propagata.
- Table **getTable**()
  - restituisce la tabella delle probabilita' del nodo

Api di Hugin: metodi utili (4)

- Per ottenere il calcolo della propagazione di evidenze nella rete usiamo:
- Node.selectState(state) setta l'evidenza sullo stato *state*
- dominio.compile(); compila il dominio con le evidenze impostate
- dominio.propagate(dominio.H_EQUILIBRIUM_SUM, dominio.H_EVIDENCE_MODE_NORMAL); propaga le evidenze.

Api di Hugin: stampa di BN

- Per stampare i risultati della propagazione e' necessario scandire tutto il grafo e ottenere i belief di tutti gli stati di tutti i nodi:
- double **getBelief**(int state) restituisce il valore dello stato *state*
- Bisogna costruire un ciclo partendo da un nodo analizza tutti gli altri

Api di Hugin: esempio stampa di BN

```
public void StampaBN(Domain domain) {
try {
    Node node;
    ListIterator it = domain.getNodes().listIterator();
    String vettore[];
    String indice1[];
    ...
node = (Node)it.next();
while(it.hasNext()) {
...
If (( node.getName().equals("nomeNodo1"))| …) {
...
}
for (x=0;x<=lunghezzaVettore;x++) {
    System.out.println(indice1[x]+": "+ vettore[x]);
    }
 } catch (ExceptionHugin e) {
 System.out.println("Exception caught:");
 System.out.println(e.getMessage());
} } }
```

Api di Hugin: esempio stampa di BN

```
If (( node.getName().equals("nomeNodo1"))|          …          |
                    … |( node.getName().equals("nomeNodoN")) {
  if (node.getKind() == Domain.H_KIND_DISCRETE) {
        indice1[x]=node.getName();
        for (int i = 0; i < ((DiscreteChanceNode)node).getNumberOfStates(); i++) {
                if (max_stato<((DiscreteChanceNode)node).getBelief(i)) {
                        max_stato=((DiscreteChanceNode)node).getBelief(i);
                        stringa= ((DiscreteChanceNode)node).getStateLabel(i);
                    vettore[x]=stringa;
                }
        }
        x++;
    }
}
```


Esercizio: Formalizzare il seguente ragionamento incerto

D crede che le ragazze siano poco portate per la matematica.

Crede che chi è poco portato per la matematica abbia difficoltà a capire la trigonometria, a meno che non abbia dato prove precedenti di bravura.

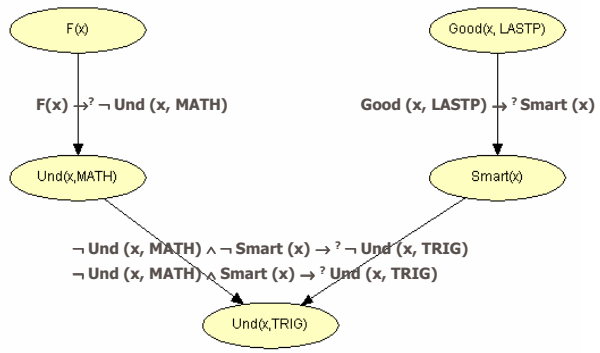
Nella sua classe, D ha una ragazza, Maria, che ha preso 8 all'ultimo compito.

D ha fatto, oggi, una lezione di trigonometria (in particolare, ha spiegato i concetti di seno e di coseno).

Vuole rendersi conto se Maria ha capito questo concetto, prima di passare a fare un esempio.

Formalizziamo la Conoscenza Generale

La rete rappresenta la *conoscenza generale*; le probabilità condizionate rappresentano i gradi d'incertezza sulle relazioni fra i diversi elementi.

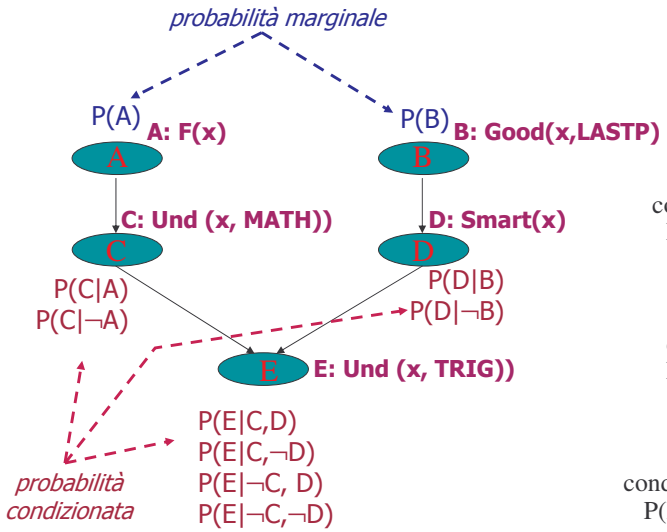


Le probabilità dei nodi-radice vengono definite sulla base delle conoscenze sulla popolazione a cui il modello viene applicato.

La *conoscenza specifica* viene rappresentata inserendo 'evidenza' su uno o più nodi. Ad esempio, ragioniamo su Maria: P(Gender(x)=F) = 1; P(Good(x,LASTP) = 1;

I quesiti (goal) vengono testati propagando l'evidenza nella rete e osservando il valore di probabilità del nodo-goal; ad es: P(Und(x,TRIG))=?

Rappresentazione con una RCP



Indipendenza condizionale tra le variabili associate ai suoi nodi:

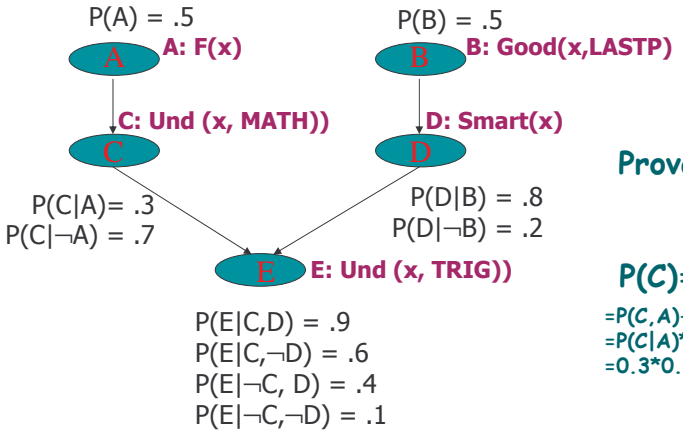
C è indipendente da B, D ed E, condizionatamente ad A: P(C|A,B,D,E) = P(C|A)

D è indipendente da A, C ed E, condizionatamente a B: P(D|A,B,C,E) = P(D|B)

E è indipendente da A e B, condizionatamente a C e D: P(E|A,B,C,D) = P(E|C,D)

Assegniamo i parametri (idea soggettiva)

Assegniamo le probabilità marginali e condizionate sulla base della conoscenza soggettiva del dominio e, quindi, del proprio 'grado di fiducia' sull'associazione fra le variabili.

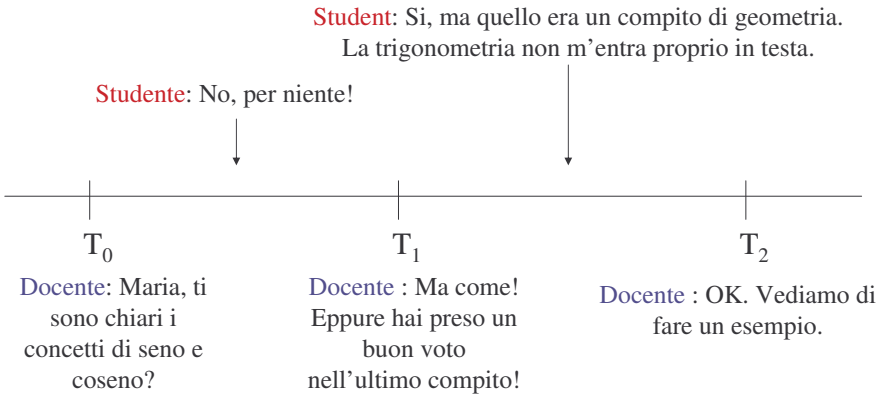


Prova a calcolare P(C) e P(E)

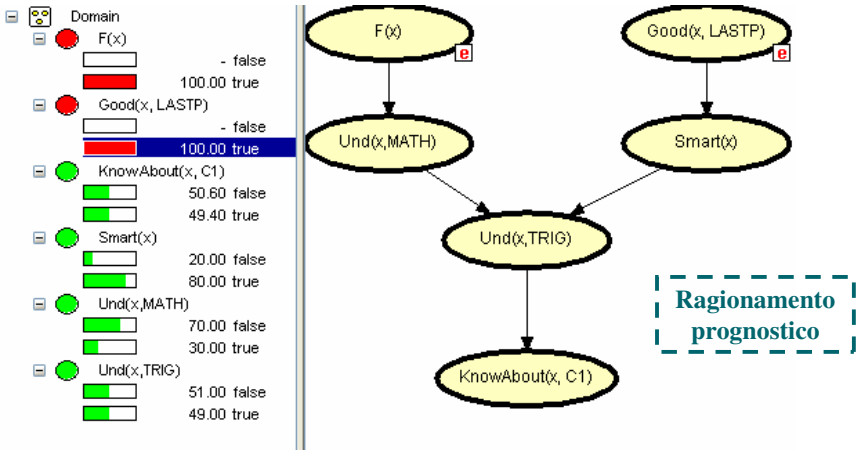
P(C)=???

=P(C,A)+P(C,¬A)=
=P(C|A)*P(A)+P(C|¬A)*P(¬A)=
=0.3*0.5+0.7*0,5=0.5

Divertiamoci con Hugin: Un Dialogo fra Docente e Studente

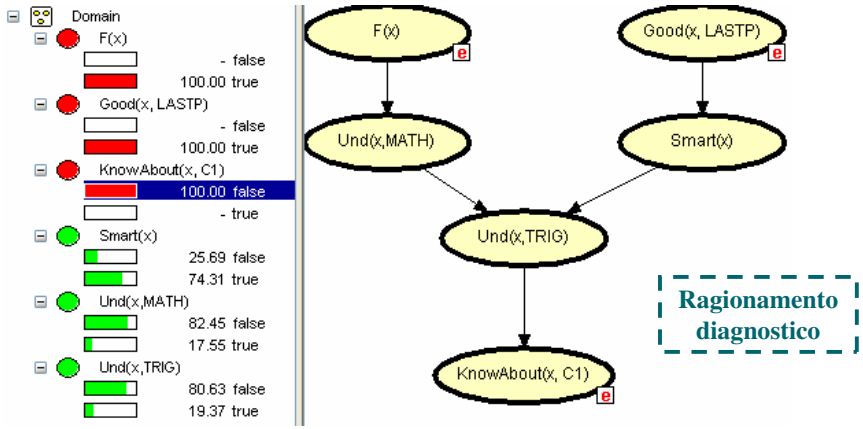


Tempo T1



D sa che Maria ha fatto bene l'ultimo compito; propaga questi dati nel suo modello di Maria. Deduce che probabilmente è brava. Ma è una ragazza,.. quindi D non è sicura che conosca i concetti di trigonometria che sta spiegando.
Chiede: *"Maria, ti sono chiari i concetti di seno e di coseno?"*
Maria: *"No, per niente!"*

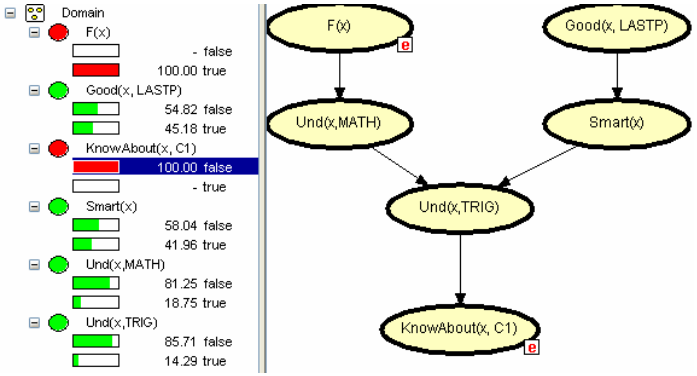
Tempo T2



D inserisce la nuova evidenza e deduce (osservando i nuovi valori di prob dopo propagazione e confrontandoli con i precedenti) che Maria, è un po' meno brava e capisce la trigonometria meno di quanto lei credesse:
Chiede: *"Ma come! Eppure hai preso un buon voto nell'ultimo compito!"*
Maria: *"Si, ma quello era un compito di geometria. La trigonometria non m'entra proprio in testa."*

Tempo T3

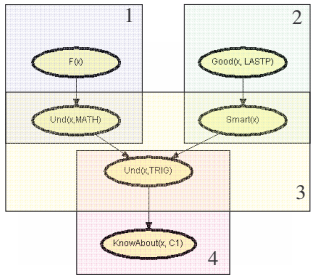
D capisce la ragione del suo errore:
ritratta il 'Good(x, LASTP) e si rende conto che, prima di passare ad un esercizio, è bene che chiarisca di nuovo a Maria i concetti di seno e coseno.



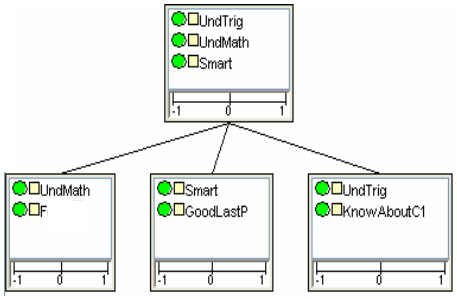
Afferma: *"OK. Vediamo di riprendere questi concetti, allora."*

...uno sguardo al file .hlg

Quali le cliques?



E lo Junction Tree?



Problema

Dopo quest'esercizio Maria avrà capito meglio i concetti di seno e coseno?

Come può il docente monitorare il grado di apprendimento di Maria?

Rappresentiamo
l'esempio con un
Belief Network Dinamico (DBN)

Belief Network Dinamici

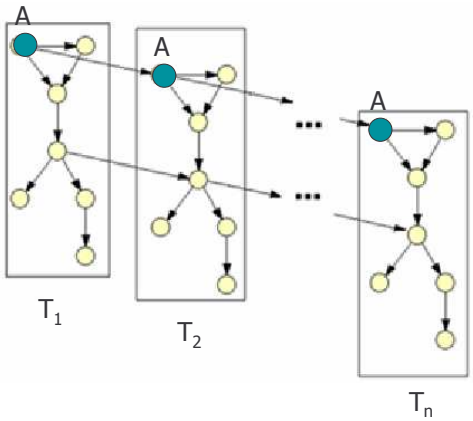
- Metodo per rappresentare conoscenza in domini "inerentemente incerti"
- Natura statica delle applicazioni: *ogni variabile è osservabile una sola volta*.

} Belief Network Statici

- Metodo per rappresentare domini che comportano anche *osservazioni ripetute di variabili*
- Comporta la connessione fra istanze multiple di belief network statici

} Belief Networks Dinamici

Modelli dinamici 'con effetto di trascinamento'



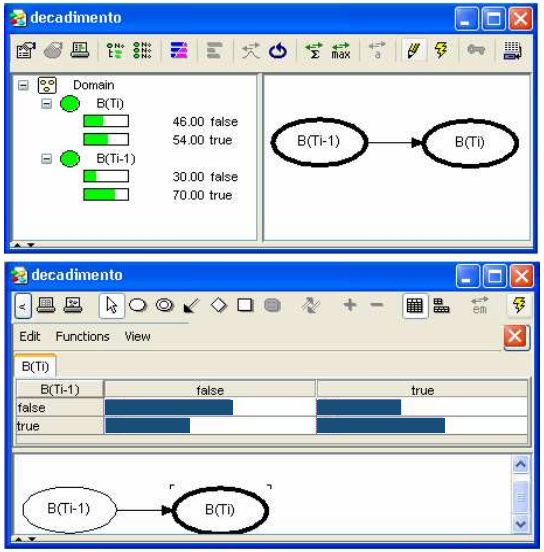
La distribuzione di probabilità della variabile associata al nodo A dipende (oltre che dalle distribuzioni delle variabili associate agli altri nodi) anche dalla distribuzione di A al tempo precedente.

Si può simulare:

- *l'effetto decadimento:*
  $Prior(A, T_i) < Prior(A, T_{i-1})$

- *l'effetto incremento:*
  $Prior(A, T_i) > Prior(A, T_{i-1})$

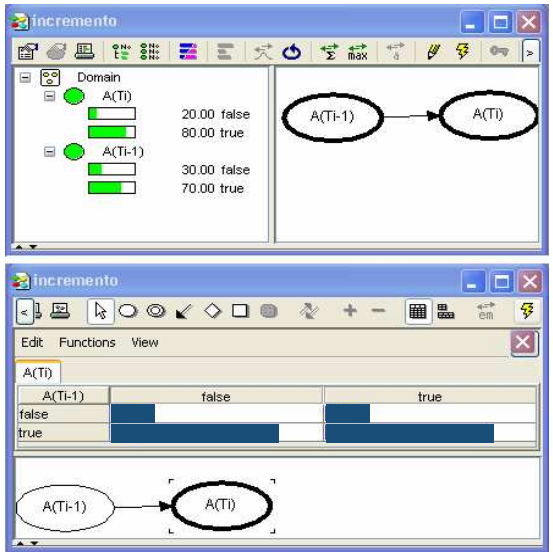
Un esempio di simulazione dell'effetto di decadimento



Ad esempio:
ad A posso associare una variabile che rappresenta lo 'stato emotivo' di U. L'ipotesi è che le emozioni tendano a decadere, nel tempo, a meno che non intervengano nuovi fattori emotivi.

Variando i due valori di probabilità condizionata, si può ottenere un effetto di decadimento più o meno forte

Un esempio di simulazione dell'effetto di incremento



Ad esempio:
al nodo B posso associare una variabile che rappresenta il grado di coinvolgimento in un gioco da parte di U.
In questo caso, l'ipotesi è che il grado di coinvolgimento cresca nel tempo (anche se possono intervenire, in ogni istante, nuovi fattori che ne cambiano il valore).

Linee guida per la creazione di DBN

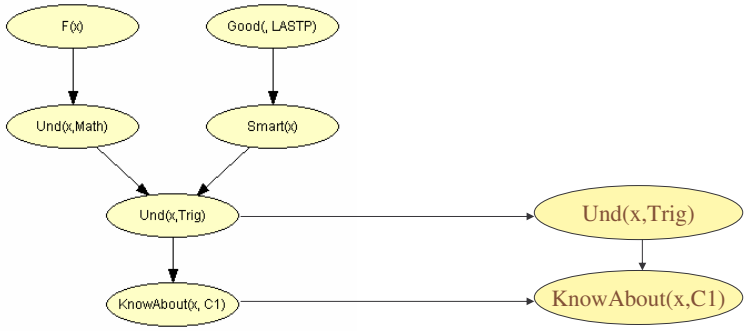
- Definire lo strato iniziale
- Definire il generico strato
- Definire le variabili dinamiche
- Definire le modalità di espansione e riduzione del modello: ad ogni mossa di U (al tempo T), aggiungiamo una fascia T+1 e eliminiamo la fascia T-1

Nel nostro esempio abbiamo già definito il generico strato e lo strato iniziale.

Quali sono le variabili dinamiche?

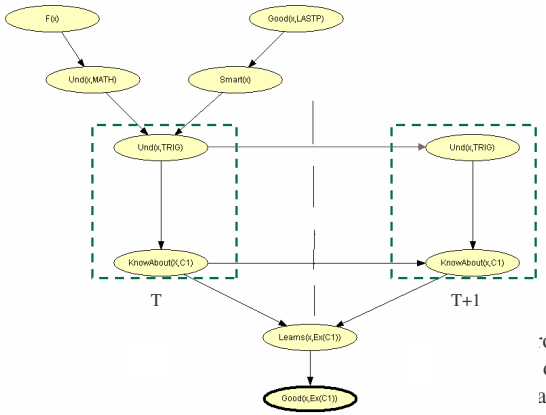
Quali sono le variabili dinamiche?

Assumo che la conoscenza della trigonometria da parte dello studente aumenti man mano che apprende nuovi concetti, attraverso spiegazioni o esempi descritti dal docente.



Inserisco un link di trascinamento (con 'incremento' della prob) fra i nodi Und(x,Trig) in due fasce di tempo consecutive.
Idem per KnowAbout(x,C1)

La truttura del DBN: cosa rappresenta?



Avevano lasciato Maria con enormi dubbi su sui concetti di seno e coseno (*Passo T*).

Il docente decide di procedere con nuovi esempi.

Supponiamo che nell'intervallo di tempo (T, T+1), Maria abbia imparato un esercizio su seno e coseno.

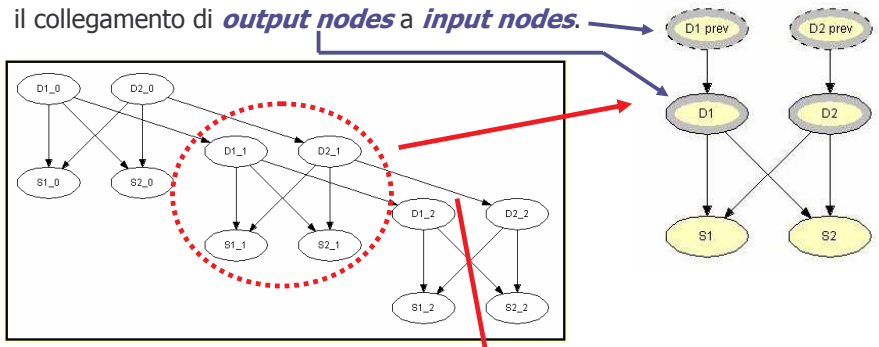
Questa evidenza va ad aumentare robabilità che Maria abbia imparato di più su seno e coseno. Il docente, ato questo e, tenuto conto del grado di conoscenza di Maria all'istante T, deduce che il grado di conoscenza di Maria circa seno e coseno è aumentato al passo T+1.

Belief Network Dinamici (Hugin Web site)

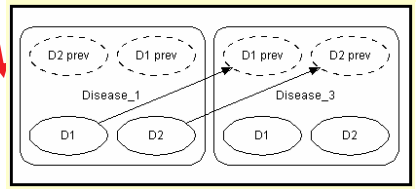
- An *Object-Oriented Network* is a network (i.e., Bayesian network or influence diagram) that, in addition to the usual nodes, contains ***instance nodes***.
- An ***instance node*** is a node representing an instance of another network. In other words, an instance node represents a subnet. Therefore, following standard object-oriented terminology, an object-oriented network is often referred to as a *class*.
- Of course, the network of which instances exist in other networks can itself contain instance nodes, whereby an object-oriented network can be viewed as a hierarchical description (or model) of a problem domain.

DBN come BN gerarchici in Hugin

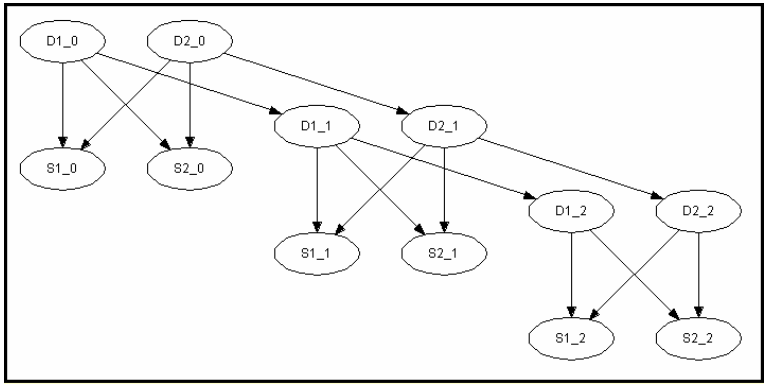
In Hugin, il collegamento fra BNs a diversi livelli di astrazione è realizzato attraverso la creazione di ***instance BN*** e il collegamento di ***output nodes*** a ***input nodes***.



Il generico strato temporale corrisponde ad un BN di livello inferiore.
Gli input e gli output nodes corrispondono alle istanze di una variabile in due strati adiacenti.
Il link fra input e output nodes rappresenta i legami fra gli strati.



Costruiamo una DBN: obiettivo

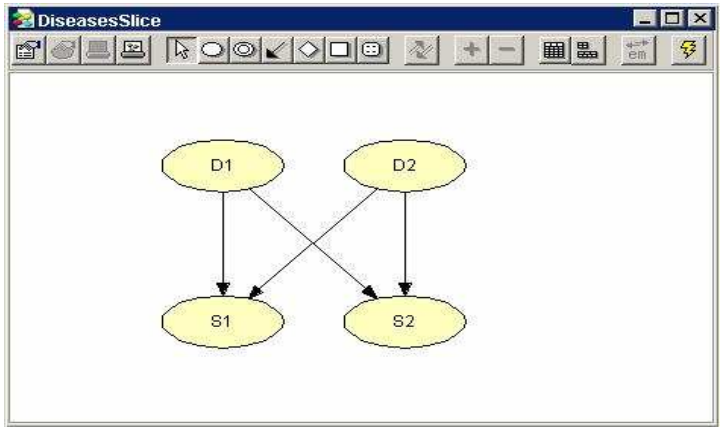


Rappresentazione mediante BN del Diseases problem.

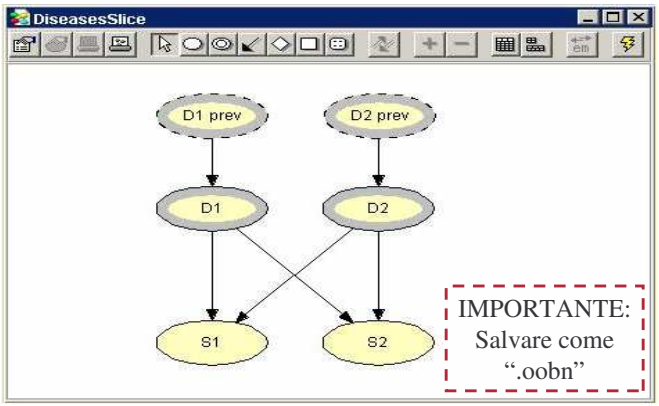
D1 e D2: due differenti malattie

S1 e S2: i sintomi che possono essere osservati come conseguenza di entrambe le malattie.

Costruiamo una DBN: Creazione della singola sottorete (time slice)



Costruiamo una DBN: Creazione dell'Interface Node



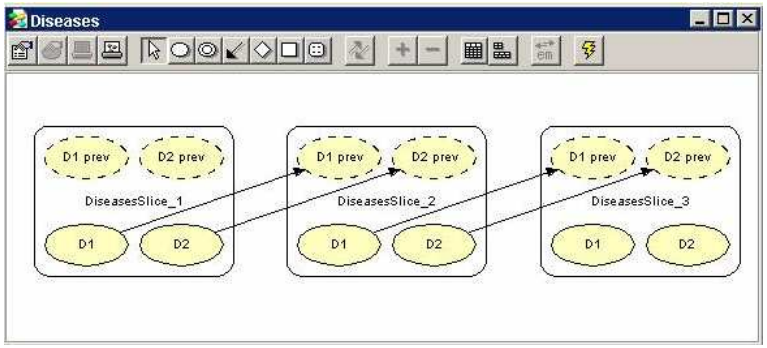
D1 e D2: *Output node* ottenuti spuntando l'"Output" check box nel Node Properties pane per ognuno di essi

"D1 prev" e "D2 prev" : *input nodes*, ottenuti creando due nuovi nodi e spuntanto l' "input" check box nel Node Properties Pane per ognuno di essi.

Attenzione: i nodi "D1 prev" and "D2 prev" sono *placeholder nodes* per D1 and D2, rispettivamente, nella sottorete immediatamente precedente i placeholder nodes sono *input nodes*, e non devono essere confusi con i nodi reali!!!

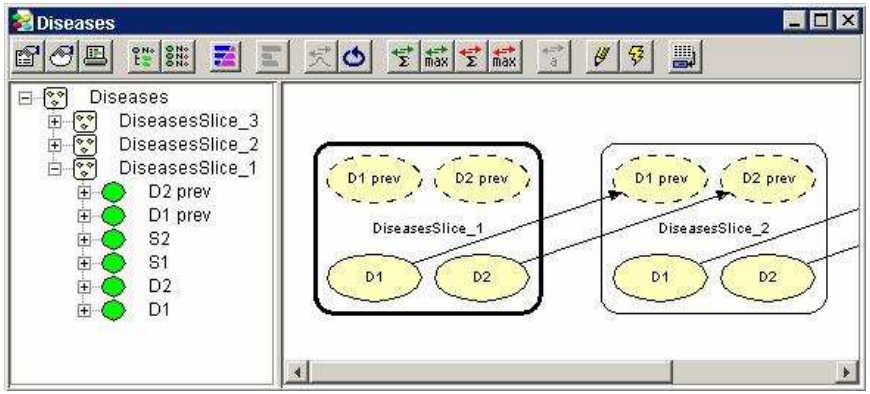
Costruiamo una DBN: Creazione del Diseases model

1) Creare una nuova rete vuota
2) Selezionare Instance Tool e creare tre instance nodes nel network pane
3) Linkare gli *output nodes* degli istance node al tempo precedente con gli *input nodes* degli instances node al tempo successivo



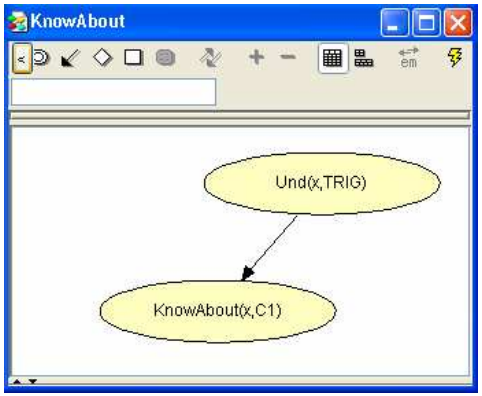
Costruiamo una DBN: Running the Object-Oriented Network

Tutto procede come fosse una normale Bayesian network!



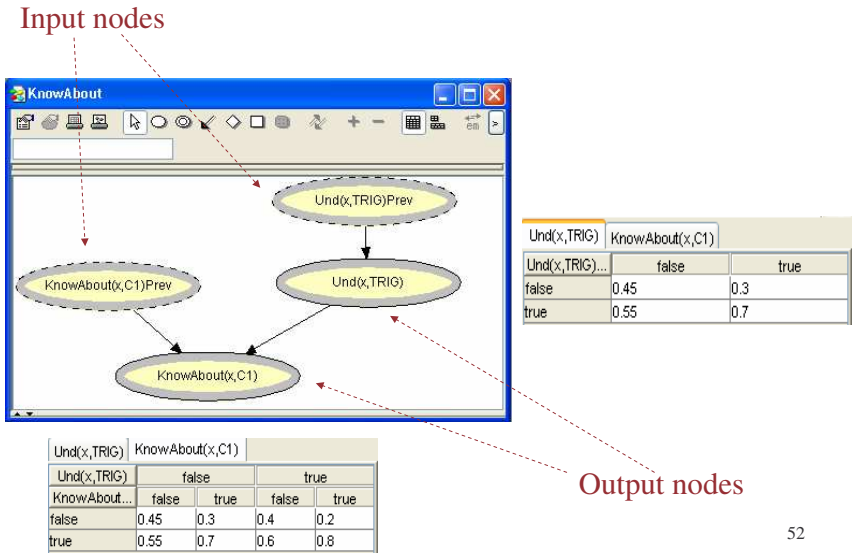
Divertiamoci con Hugin: creiamo il nostro DBN per monitorare l'apprendimento di Maria

Creazione della singola sottorete (time slice)



Creazione dell'Interface Node

(salvare come .oobn)



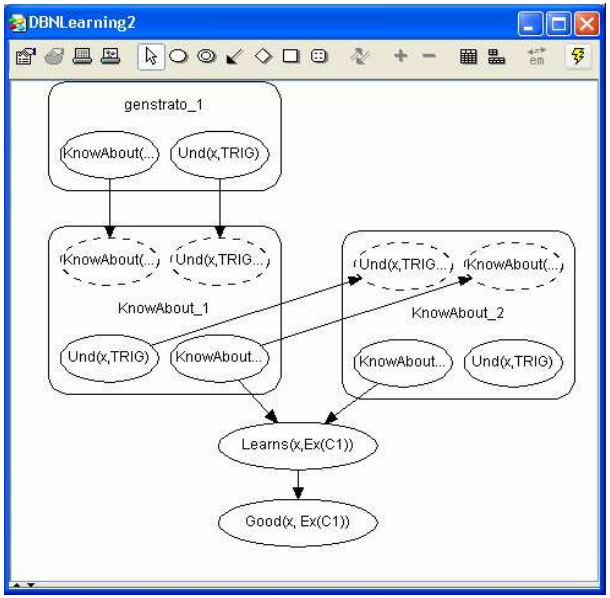
Und(x,TRIG) | KnowAbout(x,C1)

Und(x,TRIG)...	false	true
false	0.45	0.3
true	0.55	0.7

Und(x,TRIG) | KnowAbout(x,C1)

Und(x,TRIG)	false		true	
KnowAbout...	false	true	false	true
false	0.45	0.3	0.4	0.2
true	0.55	0.7	0.6	0.8

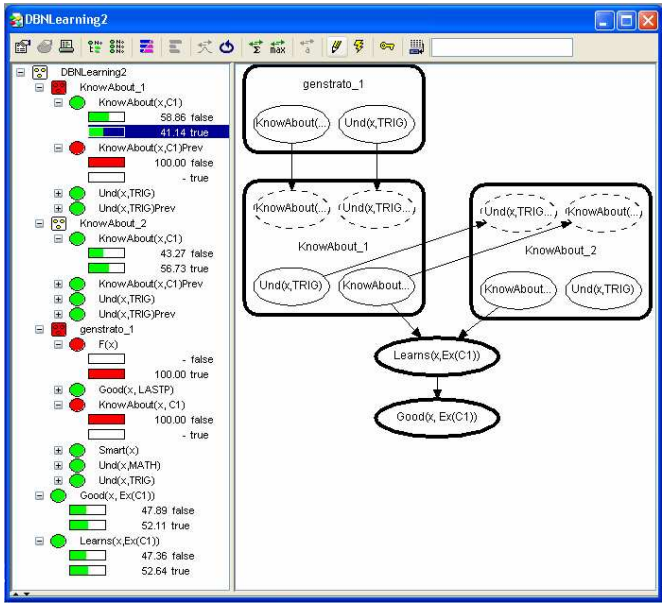
Creazione del DBN Learning Model



Learns(x,C1)	Good(x, Ex(C1))			
KnowAbout...	false		true	
KnowAbout...	false	true	false	true
false	0.9	0.4	0.4	0.1
true	0.1	0.6	0.6	0.9

Learns(x,C1)	Good(x, Ex(C1))	
Learns(x,C1)	false	true
false	0.9	0.1
true	0.1	0.9

Running the Maria's DBN Learning model



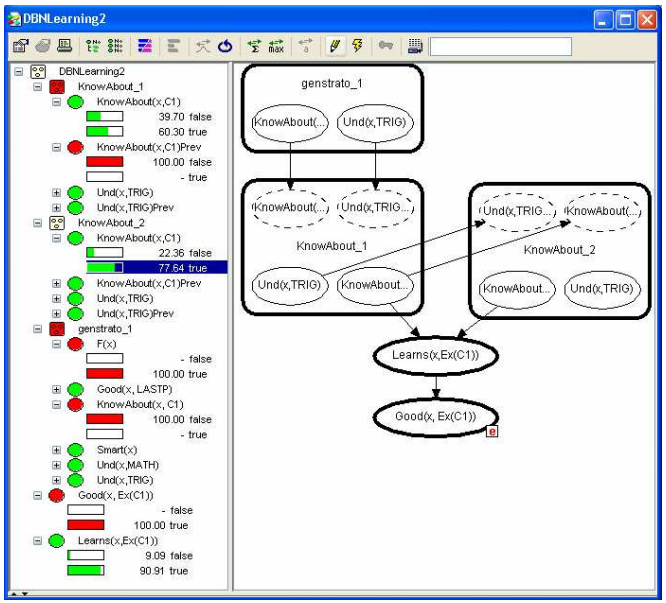
Situazione iniziale (tempo T):

Maria non ha capito i concetti di seno e coseno.

Nell'intervallo (T, T+1):

Il docente procede con un nuovo esempio

Running the Maria's DBN Learning model



Evento nell'intervallo (T, T+1):

Maria ha svolto un buon esercizio su seno e coseno

Situazione al tempo T+1:

Il docente deduce che il grado di conoscenza di Maria circa seno e coseno è aumentato.

OOBN HUGIN API
Fasi di creazione:

Progettare la rete con Hugin e salvarla rete in formato OOBN (salvare la rete prima della propagazione).

- Preparare la classe che accederà ed elaborerà le reti SEGUENDO QUESTE FASI:
  - Definire un ClassCollection
    (ClassCollection cc = new ClassCollection())
  - Richiamare il metodo parserClasses di cc per leggere i file oobn (cc.parserClasses ("nomefile.oobn")).

A questo punto le reti opportunamente convertite in classi sono presenti nel cc il quale eredita i metodi della classe vector e va gestito come un vettore.

OOBN HUGIN API
Fasi di elaborazione:

Per usare le reti precedentemente create occorre definire un dominio in cui queste reti vanno elaborate:

- Domain domain = new Domain();
- Istanziamo il dominio con domain = test.createDomain();
- domain.trinagulate (Domain.H_TM_FILL_IN_WEIGHT);
- domain.compile

A questo punto il dominio contenente le reti può essere elaborato. Infatti, è possibile riferirsi ad un qualunque dei nodo della rete per settare una evidenza; oppure leggerne un valore o modificarne il peso.

Terminata la preparazione delle reti possiamo propagare:

domain.propagate (Domain.H_EQUILIBRIUM_SUM,
Domain.H_EVIDENCE_MODE_NORMAL)

Dopo di che possiamo interrogare i nodi per osservare il risultato della propagazione

OOBN dinamiche

Obiettivo: collegare due o più reti oobn ottenendo una struttura ad albero complessa. In più, rispetto a quanto detto finora, è l'identificazione dei nodi di collegamento: nodi di Input e nodi di Output.

Linee guida per la creazione di DBN:

- Inserire tutte le classi relative alle reti oobn in un vettore di classi hugin definito in questa maniera:
  COM.hugin.HAPI.Class[ ] vetclass = new COM.hugin.HAPI.Class[10].
  L'inserimento avviene creando l'oggetto *ClassList cL = cc.getMembers()* la cui navigazione permette di prelevare ogni sua classe per inserirla nel vettore.
- Identificare: nella rete "father" il nodo di output (chiamato *actualNode*) e nella rete figlio il nodo di input (chiamato *formalNode*)
- Creare un *IstanceNode* usando la rete figlio ed inserendola nel class collection *home*:
  IstanceNode instnode = new IstanceNode (home, 'elemento del vettore delle classi contenente la rete figlio)
- Settare l'input: *instnode.setInput(formalNode, actualNode)*

Nel caso di strutture più complesse il processo va ripetuto.
Ora possiamo istanziare il dominio come visto prima ed otterremo il nostro DBN

Riferimenti

- **http://www.hugin.com/**
- **http://www-2.cs.cmu.edu/~javabayes**
- **http://www.norsys.com/**
- **http://www.lumina.com/**