Reti Bayesiane: Inferenza

Intelligenza Artificiale B
Prof. Alfonso E. Gerevini

Classi di Variabili

- Variabili di *query* (dato un *evento osservato*)
- Variabili di *prova/evidenza*: **E:** E1…Em
- Variabili non di prova (*nascoste*): **Y:**Y1…Yk
- Evento osservato **e**: insieme di valori per **E**
- Tipicamente ho una sola variabile di query (X) e un insieme di variabili prova
- Sono interessato a calcolare P(X|**e**):

  ad es. P(Furto=true|Tel-J=true,Tel-M=true)

Inferenza

Calcolare la probabilità a posteriori di un insieme di variabili di query dato il valore esatto di alcune variabili prova (evidenza)

Quattro tipologie

- Inferenza *Diagnostica*, es. P(Furto | Allarme)
- Inferenza *Causale*, es. P(Tel-Mary | Furto)
- Inferenza *Intercausale*, es. P(Furto | Allarme, Terremoto)
- Inferenza *Mista*
  - Diagnostico causale, es. P(A | T-M, Terremoto)
  - Intercausale e diagnostica, es. P(Furto | Tel-Mary, Terr.)

Inferenza per Enumerazione

- P(X|**e**) = P(X,**e**)/P(**e**) = α P(X,**e**)
- P(X|**e**) + P($\neg$X|**e**) = 1 = P(X,**e**)/P(**e**) + P($\neg$X,**e**)/P(**e**)

  => α (P(X,**e**)+ P($\neg$X,**e**)) = 1 con $\alpha = 1/$ P(**e**)

  => $\alpha = 1/$ (P(X,**e**)+ P($\neg$X,**e**)

  => posso calcolare α dopo aver calcolato P(X,**e**) e P($\neg$X,**e**)
- P(X|**e**) = α P(X,**e**) $= \alpha \sum_y$ P(X,**e**,y)

  marginalizzazione di P(X,**e**) con Y

Inferenza per Enumerazione

Esempio Furto/intrusione

Notazione: t-j: T-J=true,t-m: T-M=true, f: F=True, t: valore diTerremoto, a: valore di Allarme

$$P(F|t\text{-}j,t\text{-}m) = \alpha\, P(F,t\text{-}j,t\text{-}m) = \alpha \sum_t \sum_a P(F,t,a,t\text{-}j,t\text{-}m)$$

Sfruttando la semantica della (struttura) della rete:

$$P(f\ |t\text{-}j,t\text{-}m) = \alpha \sum_t \sum_a P(f)\ P(t)\ P(a|f,t)\ P(t\text{-}j|a)\ P(t\text{-}m|a)$$

Nel caso peggiore il numero di sommatorie è O(n) e il numero dei termini è esponenziale!

Complessità Inferenza Esatta

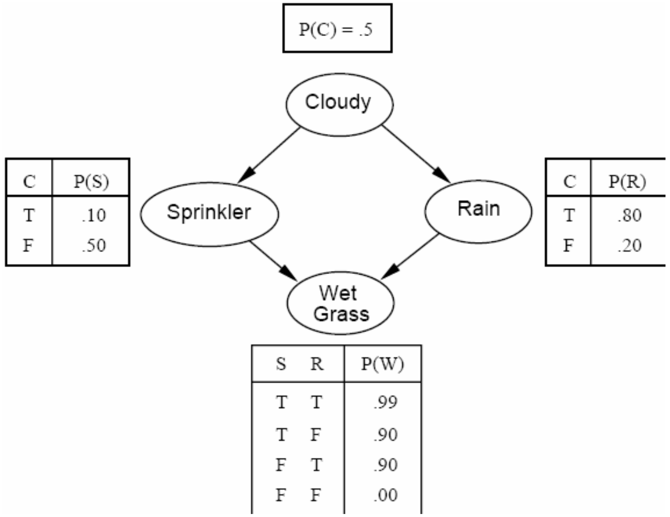
- **Eliminazione delle variabili**: è un algoritmo che consente di velocizzare i conti riducendo anche significativamente calcoli ripetuti
  - È una forma di programmazione dinamica
  - Porta vantaggi solo se la rete è un polialbero (o singolarmente connessa)
- **Reti singolarmente connesse**: *per ogni coppia di nodi X e Y esiste al più un cammino orientato oppure NON-orientato che li collega*
- **Complessità dell'inferenza esatta:**
  - Polinomiale (lineare) se la rete è singolarmente connessa
  - NP-hard nel caso generale

Inferenza nelle Reti *non* Singolarmente Connesse

- **Metodi di raggruppamento (clustering)**: trasformano la rete in un polialbero equivalente accorpando dei nodi.
- **Metodi approssimati basati su simulazione stocastica**: usando la rete generano un grosso numero di modelli.
- **Metodi di condizionamento**: trasformano la rete istanziando gruppi di variabili e valutando poi un polialbero per ogni istanziazione possibile.

Algoritmi di Clustering

Esempio "Erba bagnata" (fig 14.11 R&N)



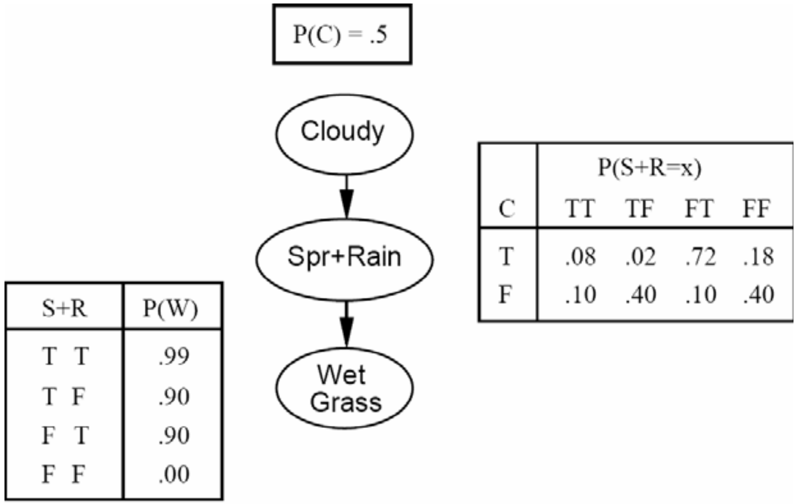
C	P(S)
T	.10
F	.50

C	P(R)
T	.80
F	.20

S	R	P(W)
T	T	.99
T	F	.90
F	T	.90
F	F	.00

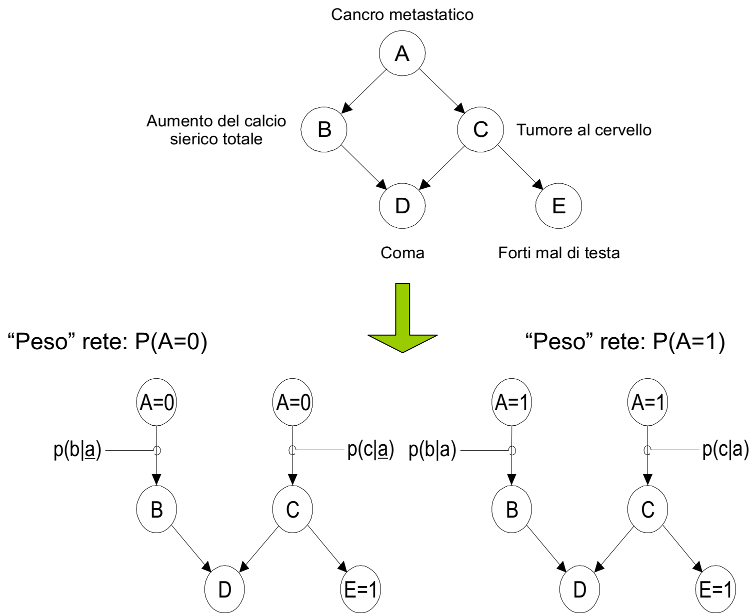
Algoritmi di Clustering

Esempio “Erba bagnata” (fig 14.11 R&N)



Metodi di Condizionamento

- Istanziazione di alcune variabili ad un valore definito (Insieme di taglio)
- Creazione di polialberi più semplici
- Calcolo delle probabilità che ci interessano su tutti i polialberi.
- Media pesata delle probabilità a posteriori (rispetto alla probabilità delle variabili condizionate)



Inferenza Approssimata: metodi di campionamento diretto

Algoritmi Monte Carlo: forniscono risposte *approssimate* la cui accuratezza dipende dal numero di campioni generati.

Campionamento casuale senza alcuna prova: campiono a turno ogni variabile, in *ordine topologico*, usando la distribuzione di probabilità condizionata ai valori già assegnati ai genitori della variabile.

Campiono P(C) = <0,5, 0,5>	…esce true
Campiono P(S | C=true) = <0,1, 0,9>	…esce false
Campiono P(R | C = true) = …	…esce true
Campiono P(W-G | I = false, P = true) = …	…esce true

C = coperto/cloudy P= piovuto I=innaffiato/sprinkler
E-B = erba-bagnata/wet grass

Evento generato dalla campionatura: **(true,false,true.false)**

Algoritmo *Campiona-Priori* (fig 14.12 R&N)

```
function PRIOR-SAMPLE(bn) returns an event sampled from the prior specified by bn
  inputs: bn, a Bayesian network specifying joint distribution P(X1, ..., Xn)

  x ← an event with n elements
  for i = 1 to n do
      xi ← a random sample from P(Xi | parents(Xi))
  return x
```

Metodo Campionamento: Calcolo delle Risposte (inferenza)

- Si calcolano *contando* ***certi*** *campioni generati*
- N: numero campioni effettuati
- **S(x1,…,xn)=**probabilità che evento x1…xn sia generato dalla campionatura
- S(x1,…,xn) = $\prod_{i:1\text{-}n}$ P(xi | genitori(Xi)) = P(x1,...,xn)
- **Ns(x1,…,xn)=** numero di eventi x1…xn generati
- Al tendere di N a infinito ci aspettiamo che

  Ns(x1,…,xn)/N = S(x1,…,xn) = P(x1,…,xn)

⇒ Per un evento parziale x1,…,xm (m < n) possiamo <u>*stimare*</u>

P(x1,…,xm) ≅ Ns(x1,…,xm)/N

Esempio: se ho 1000 campioni per la rete ErbaBagnata, e in 511 P=true, allora P(P=true) = 0,511.

Campionamento di rigetto

Può essere utilizzato per calcolare/inferire una stima di P(X|**e**)

Metodo:

1. Genera campioni dalla distribuzione a priori specificata dalla rete
2. Rifiuta tutti i campioni che non corrispondono alla prova (**e**)
3. Calcola la stima di P(X=x | **e**) contando quante volte vale X=x nei campioni rimanenti

Esempio: query = P(Pioggia | Irrigatore = true)

N = 100, 73 con I=false, 27 con I= true di cui 8 con P=true e 19 con P = false.

P(P| I = true) ≅ Normalizza (<8,19>) = <0,296, 0,704> (anziché <0,3, 0,7>)

Algoritmo Campionamento di rigetto

```
function REJECTION-SAMPLING(X, e, bn, N) returns an estimate of P(X|e)
  inputs: X, the query variable
          e, evidence specified as an event
          bn, a Bayesian network
          N, the total number of samples to be generated
  local variables: N, a vector of counts over X, initially zero

  for j = 1 to N do
     x ← PRIOR-SAMPLE(bn)
     if x is consistent with e then
        N[x] ← N[x]+1 where x is the value of X in x
  return NORMALIZE(N[X])
```

Campionamento di rigetto

- Più campioni => migliore approssimazione (la stima converge alla probabilità reale)
- **Errore**: la deviazione standard dell'errore con n campioni in ogni probabilità è proporzionale a 1/sqrt(n)
- **Problema del metodo**: servono molti campioni, che non sono facili da ottenere (molti sono rigettati!)
- La parte dei campioni generati consistente con **e** *diminuisce esponenzialmente* al crescere del numero di variabili di prova
- => il metodo non è applicabile a problemi complessi di grandi dimensioni.

Pesatura di verosimiglianza

- Evita problema campionamento di rigetto: *vengono generati solo campioni consistenti con le prove*
- I valori (osservati) delle variabili di prova **E** sono tenuti fissi e si campiona le restanti variabili X e **Y**
- Prima di contare i campioni per ogni valore possibile della variabile di query X, ogni evento è *pesato dalla sua verosimiglianza date le prove*
- **Idea intuitiva**: *gli eventi in cui è improbabile che le prove siano verificate devono pesare di meno nel conteggio*
- La verosimiglianza si calcola moltiplicando le probabilità condizionate di ogni variabile di prova, dati i sui genitori

Pesatura di verosimiglianza

Esempio rete ErbaBagnata:

<u>Query</u> P(Pioggia | Irrigatore=true, ErbaBagnata=true)

1. w = 1,0
2. Campiono P(Coperto) = <0,5, 0,5> … esce *true*
3. w = w * P(Irrigatore=true | Coperto=true) = 0,1
4. Campiono P(Pioggia|Coperto=true) = <0,8, 0,2>…esce *true*
5. w = w * P(ErbaBagnata=true | Irrigatore=true,Pioggia=true) = 0,099

⇒ Ho generato il campione (true,true,true,true) con peso 0,099 che viene conteggiato per *Pioggia = true usando 0,099 al posto di 1. Ma perché ho ottenuto un valore così basso?*

Algoritmo *Pesatura-Verosimiglianza*

function LIKELIHOOD-WEIGHTING(X, **e**, bn, N) **returns** an estimate of $P(X|\mathbf{e})$
  inputs: X, the query variable
    e, evidence specified as an event
    bn, a Bayesian network
    N, the total number of samples to be generated
  local variables: **W**, a vector of weighted counts over X, initially zero

  for j = 1 to N **do**
    x, $w \leftarrow$ WEIGHTED-SAMPLE(bn)
    $\mathbf{W}[x] \leftarrow \mathbf{W}[x] + w$ where x is the value of X in **x**
  return NORMALIZE($\mathbf{W}[X]$)

---

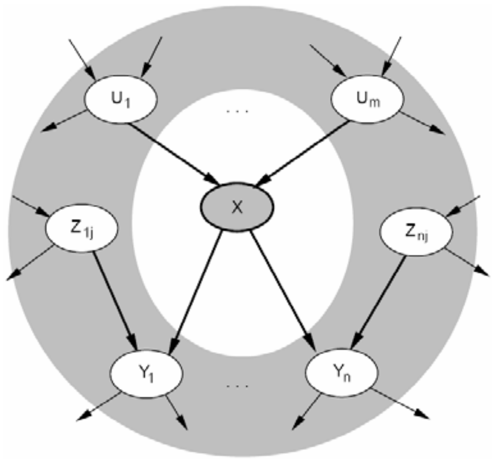
function WEIGHTED-SAMPLE(bn, **e**) **returns** an event and a weight

  $\mathbf{x} \leftarrow$ an event with n elements; $w \leftarrow 1$
  for i = 1 **to** n **do**
    if X_i has a value x_i in **e**
      then $w \leftarrow w \times P(X_i = x_i \mid parents(X_i))$
      else $x_i \leftarrow$ a random sample from $\mathbf{P}(X_i \mid parents(X_i))$
  return **x**, w

Simulazione con catene di Markov

- **Algoritmo MCMC**: <u>*Monte Carlo for Markov Chains*</u>
- Ogni evento è generato attraverso una modifica di quello precedente
- La rete ha uno **stato corrente** (valore di tutte le variabili)
- **Stato successivo**: campiono casualmente una delle variabili NON di prova Xi
- La campionatura usa la probabilità di Xi condizionata ai valori correnti di tutte le variabili nella <u>*coperta di Markov*</u> *di Xi*
- **Coperta di Markov di Xi**: *genitori di Xi* + *figli di Xi* + *genitori figli di Xi*
- MCMC fa una "random walk" nello spazio degli stati possibili

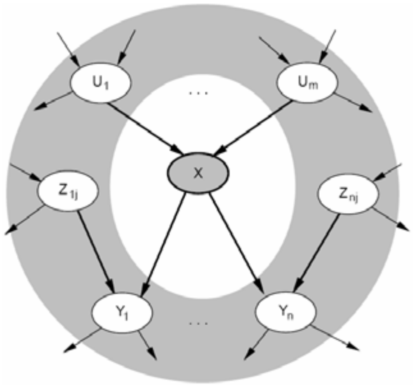
Markov Blanket (MB) per X

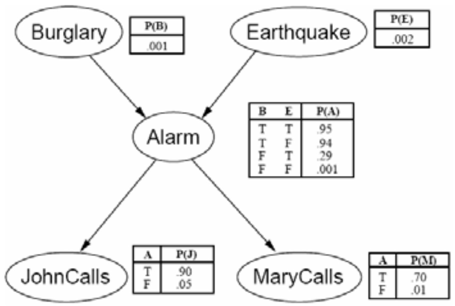


MB per X:

padri di X (nodi Ui) + figli di X (nodi Yj) + padri figli di X (nodi Zk)

MB e Indipendenza condizionale





Un nodo è condizionalmente indipendente da tutti gli altri nodi nella rete dati i nodi nella sua MB

Esempio: Bulgary indipendente da JohnCalls e MaryCalls dati Earthquake e Allarm (la MB di Bulgary)

Esempio MCMC

- Query P(Pioggia|Irrigatore=true,ErbaBagnata=true)
- Coperto e Pioggia sono inizializzate casualmente (true e false)
- Stato iniziale: **(true,true,false,true)** (= primo campione)
- Ripeti
  - Campiona <u>*Coperto*</u> con P(Coperto|Irr.=true,Pioggia=false)
    NB: **P(x|mb(X)) = α P(x | genitori(X)) * Π_{Ki} P(k | genitori(Ki))**
    Ki sono i nodi figli di X; k è il valore di Ki
    … esce *false*
    => il nuovo stato è **(false,true,false,true)**
  - Campiona <u>*Pioggia*</u> usando P(Pioggia|coperto=false,Irr.=true, ErbaBagnata=true)
    … esce *true*
    => il nuovo stato è **(false,true,true,true)**
- Risposte calcolate attraverso il conteggio dei campioni

Algoritmo *MCMC*

```
function MCMC-ASK(X, e, bn, N) returns an estimate of P(X|e)
  local variables: N[X], a vector of counts over X, initially zero
                   Z, the nonevidence variables in bn
                   x, the current state of the network, initially copied from e

  initialize x with random values for the variables in Z
  for j = 1 to N do
     N[x] ← N[x] + 1 where x is the value of X in x
     for each Z_i in Z do
        sample the value of Z_i in x from P(Z_i|mb(Z_i)) given the values of MB(Z_i) in x
  return NORMALIZE(N[X])
```

ESERCIZIO

- Estendere la rete con variabili booleane *Acquagelata* e *MotorinoAvviamento*
- Fornire ragionevoli tabelle di prob. condizionate
- La distribuzione di prob. condizionata per la var *parte* potrebbe essere descritta da una distribuzione di *AND rumoroso*. Definire questa famiglia in generale e metterla in relazione alla distribuzione di OR-rumoroso