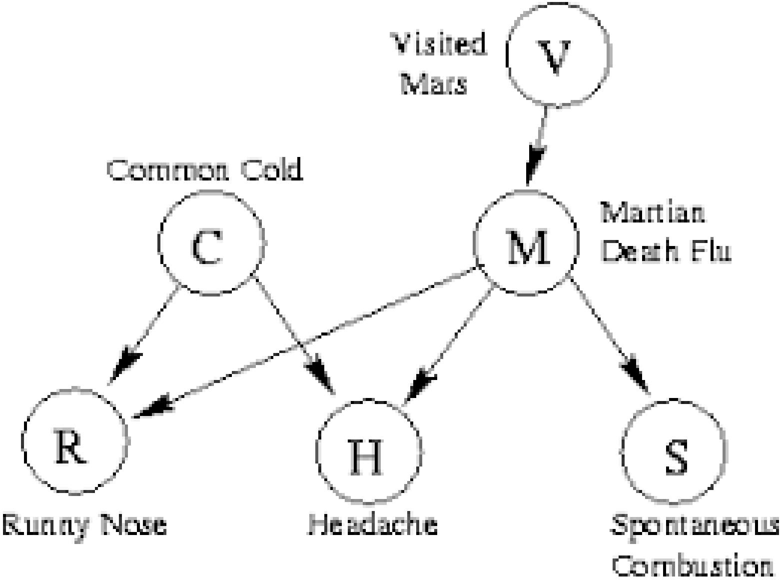
TAIA 2018/2019

Tpicos Avanados em Inteligencia Artificial

March 28, 2019

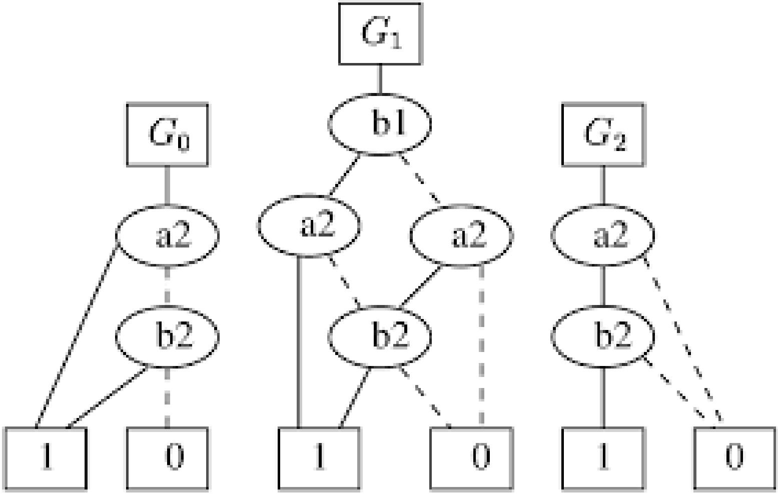
Knowledge Representation

Graphical Models



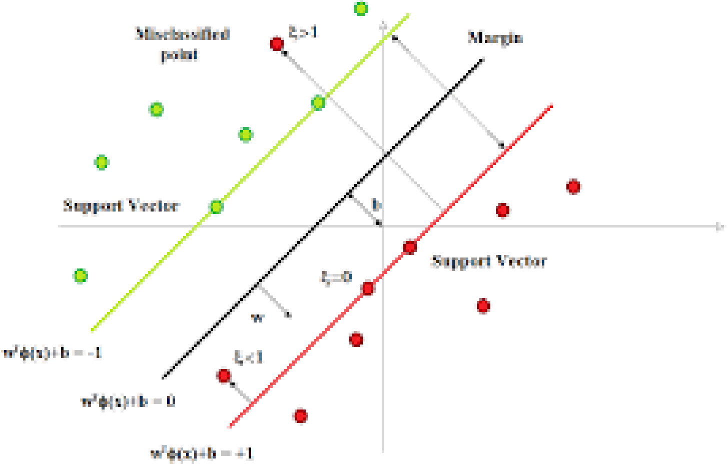
Knowledge Representation

Computational Logics



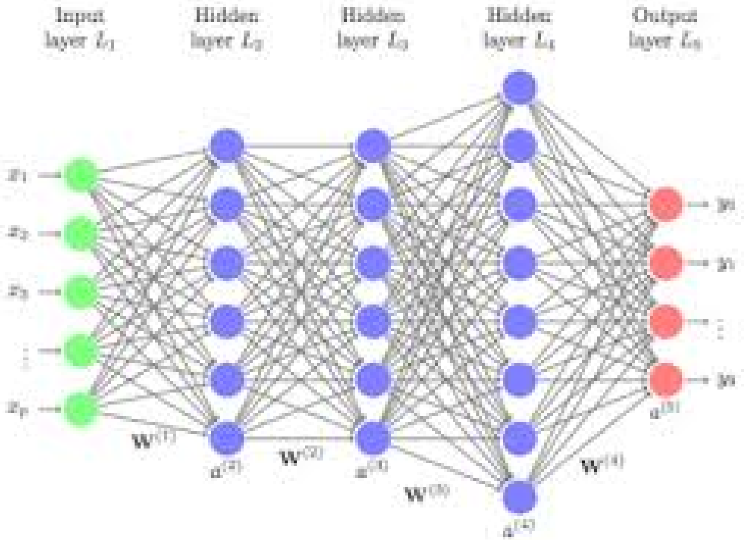
Knowledge Representation

Linear Models



Knowledge Representation

Deep Neural Networks



Probabilistic Models

- Directed Models:
  - Hidden Markov Models;
  - Bayesian Nets
  - Naive Bayes
- Undirected Models
  - Markov Networks
  - Conditional Random Fields

Key Ideas

- World is described by a set of random variables:

$$\sum Pr(X_1, X_2, \ldots, X_n) = 1$$

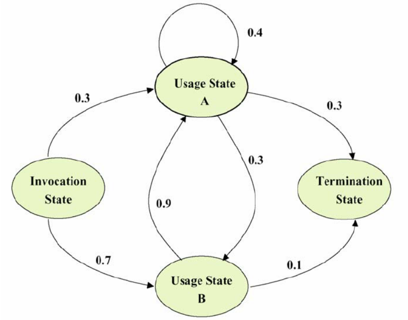
- Chain Rule:

$$Pr(X_1|X_2, \ldots, X_n)Pr(X_2|\ldots, X_n)\ldots Pr(X_n)$$

- Key Idea, *Conditional Independence*:

$$Pr(X_1|X_2, \ldots, X_n) = Pr(X_1|X_i, X_j)$$

An Example: Markov Models



$$Pr(t_0 = I, t_1 = U_A, t_2 = U_A, t_3 = U_B, t_4 = T) =$$

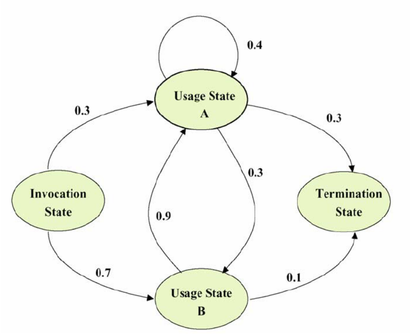
$$Pr(t_4 = T | t_0 = I, t_1 = U_A, t_2 = U_A, t_3 = U_B) =$$

$$Pr(t_3 = U_B | t_0 = I, t_1 = U_A, t_2 = U_A) \times$$

$$Pr(t_2 = U_A | t_0 = I, t_1 = U_A) Pr(t_1 = U_A | t_0 = I) Pr(t_0 = I) =$$

$$Pr(t_4 = T | t_3 = U_B) Pr(t_3 = U_B | t_2 = U_A) Pr(t_2 = U_A | t_1 = U_A) Pr(t_1 = U$$

Hidden Markov Models



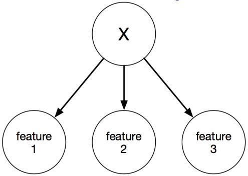
$$Pr(AAGC, EE5I) = Pr(C|AAG.EE5I)\ldots$$

$$Pr(C|I)Pr(I|5)Pr(G|5)Pr(5|E)Pr(A|E)Pr(E|E)Pr(A|E)Pr(E|Start) =$$

$$\Pi\sigma_i(o_j)\pi_{i-1\rightarrow i}$$

Viterbi: states for max $Pr(Observation, States)$.

Another Example: Naive Bayes Classifier



Classifier: $\frac{Pr(Y|X_1,X_2,X_3)}{Pr(\not{Y}|X_1,X_2,X_3)}$

$$Pr(Y|X_1,X_2,X_3) = \frac{Pr(YX_1X_2X_3)}{Pr(X_1X_2X_3)} =$$

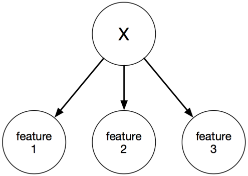
$$Pr(AAGC, EE5I) = Pr(C|AAG.EE5I)\ldots$$

$$Pr(C|I)Pr(I|5)Pr(G|5)Pr(5|E)Pr(A|E)Pr(E|E)Pr(A|E)Pr(E|Start) =$$

$$\Pi \sigma_i(o_j)\pi_{i-1\to i}$$

Viterbi: states for max $Pr(Observation, States)$.

Another Example: Naive Bayes Classifier



Classifier: $\frac{Pr(Y|X_1,X_2,X_3)}{Pr(\not{Y}|X_1,X_2,X_3)}$

$$Pr(Y|X_1,X_2,X_3) = \frac{Pr(YX_1X_2X_3)}{Pr(X_1X_2X_3)} =$$

$$\frac{Pr(X_1|YX_2X_3)Pr(X_2|YX_3)Pr(X_{|}Y)Pr(Y)}{Pr(X_1X_2X_3)} =$$

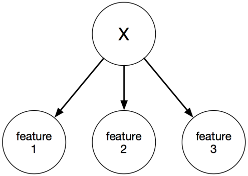
$$\frac{Pr(X_3|Y), Pr(X_2|Y), Pr(X_3|Y)Pr(Y)}{Pr(X_1X_2X_3)}$$

$$\frac{Pr(X_3|\neg Y), Pr(X_2|\neg Y)Pr(X_3|\neg Y)Pr(\neg Y)}{Pr(X_1X_2X_3)}$$

Probabilities and Logics

- Why logic:
  - Understandable Models
  - Well-defined meaning
  - Repeated Structure (first order)

At the beginning



- Propositional Logic: sentences + connectives
- Deduction: Modus Ponens, Resolution
- applied to Mathematics in the XIX/early XX Century
- many varieties: classical, intuitionistic

Probabilities and Logics

- Prove ϕ is true, given some KB Δ
- : Issues:
  - can we always prove truth/falsehood?
  - Semantic: often used Closed World Assumption
  - Technical: Some logics are undecidable (Peano's Arithmetic)
  - Inference: we may not be guaranteed to find a solution in useful time: termination, NP.

Inference in Logic

- Like in BN
  - Exact Inference: SAT Solver, Resolution
  - Approximate Inference: SAT solvers, similar to MCMC

  SAT:
  - Equivalence Checking, ie, two circuits the same?
  - Model Checking, ie, does property P hold;
  - Constraints and OR;
  - Planning (but best planners use Machine Learning, see "Delfi: Online Planner Selection for Cost-Optimal Planning"
  - Approximate Inference: SAT solvers, similar to MCMC
- Best SAT Solver also uses ML: "MapleSAT: Combining Machine Learning and Deduction in SAT solvers".

SAT Solvers

- Canonical Form: CNF

$$(a \vee b \vee \neg c) \wedge (\neg \vee \neg a \vee \neg d)$$

- Intuition: satisfy all the disjoints, *clauses*;
- Propagation:
  1. $a \wedge (a \vee b) \wedge c \rightarrow c$
  2. $a \wedge (\neg a \vee b) \wedge c \rightarrow b \wedge c$
  3. $a \wedge (\neg a) \wedge c \rightarrow$ **False**
- SAT Solvers: use these ideas to find satisfibility.

Sat Solving

- While c:
  1. Pick a α, set to `true` or `false`;
  2. Propagate
- Tricks:
  - Find the culprit:

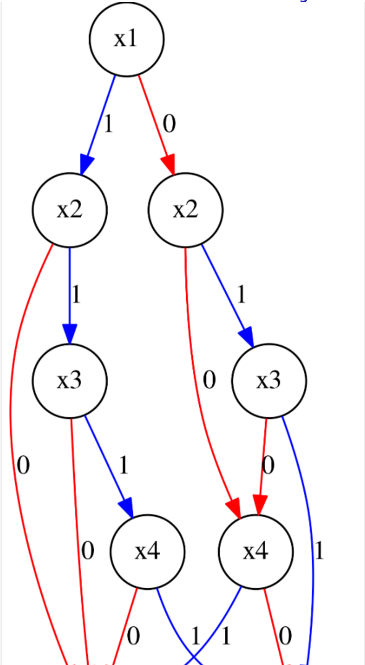
$$(\neg\alpha \vee \theta \vee \neg c) \wedge \ldots (\alpha \vee b) \ldots \wedge (\neg\alpha \vee \neg b \vee \theta) \wedge (\neg\theta \vee \neg b)$$

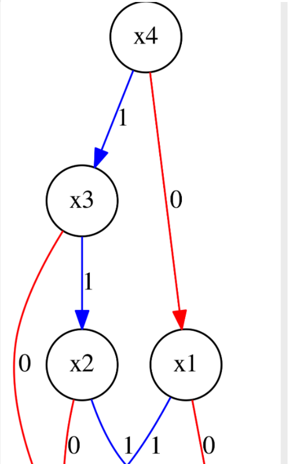
  - Set $c = 1$, $(\alpha \vee b) \ldots (b \vee \neg\alpha \vee c) \vee (b \vee \neg c)$
  - Smart Backtracking: find the root of the conflict,
  - Learning: store patterns that caused conflict
  - *Pre-Compilation:* assemble large subsets of the graph.

ODBBs: Ordered Binary Decision Diagrams

- Proposed by Edmond Clarke for symbolic model checking, eg,temporal logics;
- *while avoiding deduction*
- Each node is a boolean decision node:
  - $T = \alpha \wedge L \vee \neg\alpha \wedge R$ and
  - $T = \alpha \wedge L \vee \neg\alpha \wedge \neg R$
- Nodes always follow the same ordering from root to branch
  - The same variable may have several times, but at the same level
- No duplicated sub-trees: if T is rooted in α, there is no other T' rooted in another instance of α

ODBBs: Order is Everything





ODBBs: Great, but why care?

- Excellent for model checking simple languages
- Can they be used for KR?
- Very low-level for propositional only
- But with Probabilities:
- Imagine we know $Pr(\alpha_i)$ and that the α_i are *independent*.
- Want to know the total probability.
  - Base Cases, $Pr(1) = 1$, $Pr(0) = 0$;
  - Induction: $Pr(N_\alpha) = Pr(\alpha \wedge L \vee \neg\alpha \wedge (\neg)R)$
  - Exclusivity: $Pr(\alpha \wedge L) + Pr(\neg\alpha \wedge (\neg)R)$
  - Independence: $Pr(\alpha)Pr(L) + (1 - Pr(\alpha))Pr((\neg)R)$
- Dynamic Programming in action....

ODBBs: Great, but why care?

- ProbLog uses this method to combine Prolog rules and probabilties;
- See ProbLog-II in Leuven
- Bayesian networks can use this, but:
  - $Pr(A|BC)$ requires B and C below A, or
  - must follow a topological sort of the graph
  - also, the BDDs are pretty scary
  - People prefer ACs and their descendents...
- What about learning?

ODBBs: parameter learning

- We can do EM, because it uses DP
- More fun to use gradient descent:
- Maximize $MSR = \sum_E (Pr(E) - \overline{Pr}(E))^2$
- that is $\frac{\delta MSR}{\delta \alpha_i} = \sum_E -2 * (Pr(E) - \overline{Pr}(E)) * \frac{\delta \overline{Pr}(E)}{\delta \alpha_i} = 0$
- Going back to the DP equations, we get:
  - $i \neq j \frac{\delta \alpha_j * P_L + (1 - alpha_j) * P_R}{\delta \alpha_i} = \alpha_j * \frac{\delta P_R}{\delta \alpha_i} + (1 - \alpha_j) \frac{\delta P_R}{\delta \alpha_i}$
  - $i = j \frac{\delta \alpha_i * P_L + (1 - \alpha_i) * P_R}{\delta \alpha_i} = P_L + \alpha_i \frac{\delta P_L}{\delta \alpha_i} + (1 - \alpha_i) \frac{\delta P_R}{\delta \alpha_i} + (1 - P_R)$
- Done yet?

ODBBs: parameter learning

- We have no guarantee $0 \leq \alpha \leq 1$
- We can clamp them, ugly
- Usual trick, sigmoid function:

$$\alpha = sigmoid(\theta) = \frac{1}{1 - e^{-\theta}}$$

- Nice Derivative:

$$\frac{d\alpha}{d\theta} = \alpha(1 - \alpha)$$