Resolução de problemas por meio de estratégias de busca

Sara Ferreira
sara.ferreira@utp.edu.br

05/11/2020

1 Introdução

O presente trabalho visa sumarizar o artigo "Raciocínio Probabilístico em Sistemas Embarcados"[1]. Está organizado em 3 capítulos principais. O primeiro aborda os algoritmos utilizados pelos autores do artigo. Já o segundo capítulo, aborda o problema retratado no artigo. E o terceiro, mostra os resultados experimentais.

O artigo apresenta técnicas para utilizar o raciocínio probabilístico a sistemas embarcados. O objetivo é criar ferramentas que permitam qualquer computador embarcado manipular incertezas.

2 Fundamentação teórica

Como o objetivo é otimizar o uso da memória, o algoritmo Gibbs sampling foi implementado numa placa embarcada, a aJ-PC104. Baseado no microprocessador Java, JEM2 direct-execution, o aJ-PC104 proporciona uma eficiente plataforma para desenvolvimento de aplicações em tempo real inteiramente em Java. A placa inova ao processar diretamente as instruções de máquina da linguagem Java, sem necessidade de interpretadores.

A linguagem Java foi escolhida por ser uma linguagem com características apropriadas para sistemas embarcados. Nesses sistemas, modelos centralizados estão sendo substituídos por modelos envolvendo trabalho em rede com certo nível de inteligência e de autonomia embarcada. Ou seja, existe uma tendência atual de descentralização do processamento, potencializado pelos padrões de comunicação da Internet.

A portabilidade da linguagem Java e sua similaridade com a linguagem C, faz dela ideal para o uso em comunicação via Internet. E o atual desenvolvimento de hardware, torna possível à linguagem Java avançar além das plataformas UNIX/WINDOWS e ser inserida em sistemas embarcados.

O algoritmo Gibbs sampling foi implementado adicionando-se um método a um programa já existente, o JavaBayes, o qual é uma implementação de algoritmos de eliminação de variáveis em redes Bayesianas; o programa é distribuído livremente através de licença GNU. O programa Gibbs sampling possui a seguinte estrutura:

1. Gerar um estado inicial qualquer para as variáveis da rede.

2. Entrar com as n simulações a serem realizadas. Repetir n vezes os procedimentos abaixo.

   (a) Percorrer cada uma das variáveis.

   (b) Se a variável não estiver observada, fazer os procedimentos abaixo.

      i. Calcular a pct da variável, resultante de uma produtória. Nor-

malizar o vetor de probabilidades, ao final.

ii. Gerar uma amostra para a variável, gerando-se um número aleatório (entre 0 e 1) e comparando-o com a pct normalizada, obtida anteriormente.

iii. Se a variável amostrada anteriormente assumir o valor desejado, adicionar ao contador nx.

3. A probabilidade desejada é dada pela razão: nx/n.

A boa qualidade dos números aleatórios gerados constitui o segredo para a eficiência destes algoritmos de simulação. No Gibbs sampling implementado, empregou-se um eficiente gerador de números aleatórios, o MersenneTwister.

3 Análise dos problemas

Uma análise dos algoritmos existentes para inferências com redes Bayesianas levou à escolha de métodos aproximados por amostragem, já que estes métodos utilizam pouca memória. Redes Bayesianas são representações compactas e eficientes para um conjunto de distribuições de probabilidade. Normalmente, as redes Bayesianas são utilizadas para realizar cálculos de probabilidade condicional em um conjunto de variáveis observadas.

O caminho para realizar inferências em sistemas embarcados é utilizar aproximações. As aproximações que ocupam menos memória são baseadas em amostragem. A proposta do método Monte Carlo é que se calcule o valor das integrais da equação numericamente, ou seja, cálculos aproximados, através de amostras geradas em seu domínio de integração.

Para gerar estas amostras, é utilizado o "logic sampling" no qual deve-se ordenar os nós da rede, de maneira que os nós sempre estejam precedidos de seus pais, e amostrá-los em sequência. Assim, começando com $X1$ (um nó sem pais), encontra-se uma amostra de $X1$ a partir de $p(X1)$. Depois, seguindo para $X2$ e encontra-se uma amostra de $X2$. E assim sucessivamente, obtém-se uma amostra $(X1, X2, ..., Xn)$.

O passo a seguir é verificar se a amostra é consistente com as evidências. Se a evidência não é representada na amostra, descarta-se esta amostra; caso a evidência seja consistente com a amostra, retém-se a amostra. Parte-se então para uma nova amostragem, até que se obtenha uma que seja consistente.

O problema do "logic sampling" é que: quanto maior for o número de evidências, mais amostras devem ser geradas para que uma amostra seja aceita. Se uma das evidências têm pouca probabilidade para ocorrer, muitas amostras terão que ser geradas até que consiga uma que seja consistente.

Este tipo de problema é solucionado por métodos aproximados conhecidos como métodos MCMC (Monte Carlo Markov Chain). Esses métodos se baseiam na aplicação de técnicas de Monte Carlo em uma cadeia de Markov especialmente construída para simular o modelo de interesse.

Vários métodos utilizam essa abordagem; entre esses métodos Gibbs sampling é um método de implementação relativamente simples em redes Bayesianas. Gibbs sampling é um caso específico do algoritmo Metropolis-Hastings. A ideia básica do Gibbs sampling é utilizar a probabilidade condicional total (pct) de cada variável para gerar as amostras.

O pct representa a probabilidade de uma variável da rede assumir um certo valor, dado o estado de todas as demais variáveis. Uma das vantagens de algoritmos MCMC é que estes algoritmos podem ser codificados de forma bastante compacta e podem ocupar reduzidas porções de memória.

Os sistemas existentes hoje para redes Bayesianas possuem versões sem interface gráfica, mas a porção de memória ocupada por tais sistemas em geral é alta e não se adapta a sistemas embarcados. O desafio é portanto implementar Gibbs sampling para redes Bayesianas de uma forma que seja adequado para sistemas embarcados. O restante deste artigo descreve uma implementação especialmente feita com esse objetivo, e os testes realizados em uma placa de processamento embarcado.

4 Resultados experimentais

Os testes foram realizados no exemplo Dog Problem. A seguinte rede Bayesiana (Figura 1) representa uma situação do cotidiano: queremos visitar um amigo, e queremos saber qual a chance de sua família estar em casa. Se a família estiver em casa há uma certa possibilidade de que a luz esteja acesa e de que o cachorro esteja vigiando a casa. Mas, se o cachorro estiver doente, isto pode influenciar o fato do cachorro estar em vigia. E, se o cachorro está vigiando, existe uma possibilidade de que ele esteja latindo.

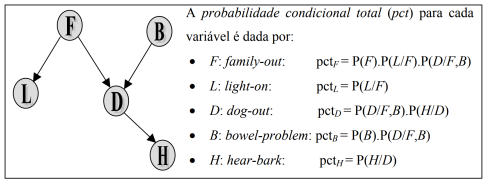

Figura 1: Representação do Dog Problem

Este exemplo é um dos exemplos contidos no pacote do programa JavaBayes. Os resultados obtidos no programa Gibbs sampling(Gibbs) são apresentadas na Figura 2, juntamente com os resultados obtidos no JavaBayes, que realiza cálculos exatos. A probabilidade P(F/L,H) é a chance de que a família esteja fora de casa, sendo que a luz está apagada e se ouvem latidos de cachorro.

F	L	H	P(F=*true*/L=*false*,H=*true*) (*Java*Bayes)	*Gibbs* n=100	*Gibbs* n=1000	*Gibbs* n=10.000	*Gibbs* N=20.000
true	*true*	*true*	0,8578	0,88	0,866	0,8572	0,86065
true	*true*	*false*	0,5006	0,67	0,513	0,5103	0,50864
true	*false*	*true*	0,1748	0,25	0,194	0,1798	0,17489
true	*false*	*false*	0,034	0,03	0,044	0,0368	0,03367

Figura 2: Resultados do programa Gibbs sampling.

Na Figura 3, está plotado o tempo gasto no aJ-PC104 para se efetuar as simulações.

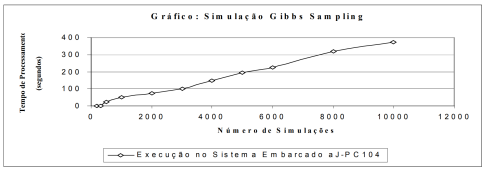

Figura 3: Desempenho do algoritmo no Sistema Embarcado aJ-PC104.

Plotando-se alguns dos resultados da Figura 1, temos o seguinte gráfico (Figura 4):

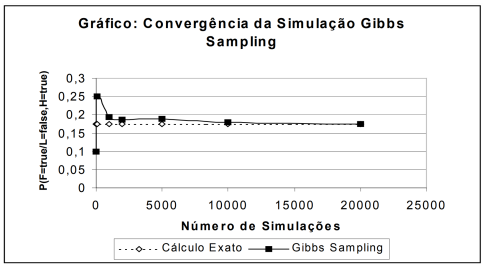

Figura 4: Comportamento da Convergência.

Os resultados obtidos foram bastante satisfatórios. Para n=2000 simulações, tem-se um desvio de 6%. Para n>10.000 simulações, tem-se desvios menores que 3%. Comparando-se com outros testes realizados, o algoritmo Gibbs sampling perde eficiência para probabilidades "limites", muito próximos de 0 ou de 1.

O tempo gasto para processamento no sistema embarcado (Figura 3) é maior que num PC (100 Mhz). Isto se deve aos recursos limitados do atual sistema (40 Mhz de velocidade e 1 Mbytes de memória RAM). É interessante

notar ainda neste gráfico, que o tempo de processamento varia linearmente com o número de simulações. Ou seja, o crescimento do tempo necessário de processamento varia linearmente de acordo com o número de iterações e independentemente da estrutura da rede Bayesiana, diferentemente do crescimento exponencial dos cálculos exatos.

5 Considerações finais

Neste trabalho foi apresentado um resumo do artigo "Raciocínio Probabilístico em Sistemas Embarcados". Para julgar os resultados obtidos, deve-se considerar que a implementação de Gibbs sampling que roda na placa aJ-PC104 ocupa apenas 4 Kbytes de memória (código executável), mais um espaço fixo para as variáveis de programa. Toda a ocupação de memória pode ser prevista no início da execução, e portanto um sistema embarcado de baixa velocidade e pouca memória (como o sistema por nós utilizado) pode empregar o algoritmo implementado sem dificuldades.

Referências

[1] Ide, J. S., and Cozman, F. G. Raciocínio probabilístico em sistemas embarcados. *Escola Politécnica, Universidade de São Paulo*.