

Algoritmos e Modelação Computacional Projecto



Paulo Mateus

Campos de Markov Aleatórios - Árvores

LEBiol – LEBiom - LMAC

2022-23

Objetivo

- O objetivo do projeto é implementar um algoritmo de aprendizagem baseado em Campos de Markov Aleatórios (Markov Random Fields)- um bloco constituinte das Deep Belief Networks (quando se usam variáveis escondidas).
- Vamos implementar o algoritmo de Chow-Liu para aprender MRFs ótimas no contexto de classificação.
- O dados Biomédicos são públicos e fornecidos na página da disciplinas e provêm do UCI machine learning repository - <http://archive.ics.uci.edu/ml/>

Classificador

Um *classificador* sobre um domínio D é simplesmente um mapa $f : D \rightarrow C$ onde C é chamado o *conjunto de classes*. Por exemplo, para o caso da base de dados *Cancer*, o conjunto de classes é $C = \{\text{benign}, \text{malignant}\}$ e um elemento em D corresponde a um tuplo de dez medições sobre o tumor. Nos casos de interesse, o domínio é sempre estruturado da seguinte forma: $D = \prod_{i=1}^n D_i$ onde n é o número de medições e D_i é o domínio da i -ésima medição. Assim, um elemento $d \in D$ é da forma $d = (d_1, \dots, d_n)$.

Dados

O classificador é construído (ou aprendido) a partir de um conjunto de dados T . Os dados são uma amostra de elementos do domínio e respectiva classe ou seja $T = \{T_1, \dots, T_m\}$ e $T_j = (d_{1j}, \dots, d_{nj}, c_j)$ onde m é a dimensão dos dados, $d_{i,j} \in D_i$, $c_j \in C$ para todo o $1 \leq i \leq n$ e $1 \leq j \leq m$. Como os dados são discretizados, isto é $D_i \subseteq \mathbb{N}$, podemos ver os dados como uma matriz $m \times (n + 1)$ de entradas naturais.

Classificar

Uma maneira simples de classificar consiste em inferir a distribuição que gera os dados (há muitas outras maneiras). Sejam $X_1 \dots X_n$ e C variáveis aleatórias para as quais os dados T são uma amostra multinomial do vector aleatório $\vec{V} = (X_1 \dots, X_n, C)$. O objectivo de classificar pode-se reduzir a inferir a distribuição deste vector da seguinte forma

$$f(d_1, \dots, d_n) = c$$

tal que $\Pr(\vec{V} = (d_1, \dots, d_n, c)) > \Pr(\vec{V} = (d_1, \dots, d_n, c'))$ para $c' \neq c$.

Por outras palavras, sabendo a distribuição do vector \vec{V} , classificar um elemento do domínio reduz-se a escolher o elemento da classe que maximiza a probabilidade de observar o elemento do domínio com este elemento da classe (ou seja f é o estimador de máxima verosimilhança para a classe dado o elemento do domínio).

Lei dos grandes números

Note que a dimensão do domínio D cresce exponencialmente com o número de variáveis, e portanto inferir a distribuição (multinomial) do vector V utilizando a lei dos grandes números¹ requer dados de dimensão exponencial no número de variáveis para obter distribuições próximas das distribuições reais. Nestas condições, quando se utilizam dados pequenos, a distribuição obtida fica muito enviesada aos dados, fenómeno a que se dá o nome de *overfitting*.

$$^1\text{Prob}(\vec{V} = (d_1, \dots, d_n, c)) = \lim_{m \rightarrow \infty} \frac{|\{i \leq m : T_i = (d_1, \dots, d_n, c)\}|}{m} \text{ e } T \text{ é uma amostra arbitrariamente grande.}$$

Vamos ter de restringir as dependências de variáveis!

Campo de Markov Aleatório

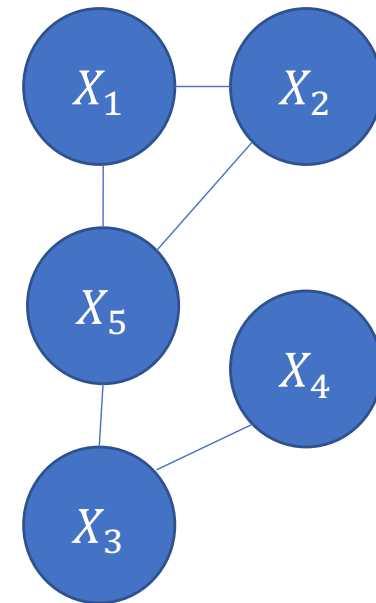
Um Campo Aleatório de Markov
ou *Markov Random Field* (MRF) é

- 1) Vetor aleatório $\mathbf{X}=(X_1, \dots, X_n)$
- 2) $G=(\mathbf{X}, E)$ onde $\mathbf{X}=\{X_1, \dots, X_n\}$
Dito o *suporte do MRF*
- 3) Seja A e B conjuntos disjuntos de nós

$$\Pr(X_A X_B | X_S) = \Pr(X_A | X_S) \Pr(X_B | X_S)$$

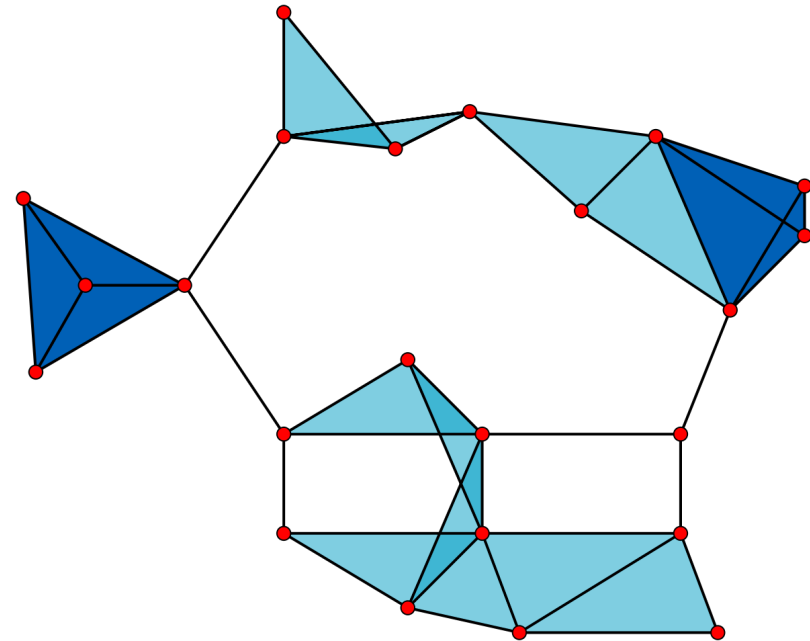
desde que para ir de um nó de A para um nó de B se passe por S em G

- 4) $\Pr(\mathbf{X}=\mathbf{x})>0$ para todo o \mathbf{x}



Clique de um grafo

Um *clique* de um grafo é um subgrafo fortemente conexo maximal.



Teorema de Hammersley–Clifford

Seja $\text{cl}(G)$ o conjunto de cliques do grafo
então

$$P(X = x) = \prod_{C \in \text{cl}(G)} \phi_C(x_C)$$

e que a distribuição é de Boltzmann

$$P(\mathbf{x}) = \frac{e^{-\text{Energy}(\mathbf{x})}}{Z},$$

$$Z = \sum_{\mathbf{x}} e^{-\text{Energy}(\mathbf{x})}$$

Dificuldade de aprender MRF

- Note que quando mais esparsos for o grafo mais assunções de independência são feitas sobre a distribuição. No caso do grafo ser totalmente esparsos (sem arestas) vamos cair no chamado *Naive Bayes Classifier* em que todas as variáveis são independentes entre si exceto com a classe
- Se o grafo for completo, estamos na situação de não ser possível obter (e até guardar) dados suficientes para aprender a distribuição!
- Não há algoritmos eficientes para aprender o melhor MRF para um conjunto de grafos esparsos. Apenas se conhece um **algoritmo eficiente para árvores!**

Algoritmo de Chow-Liu

Árvores

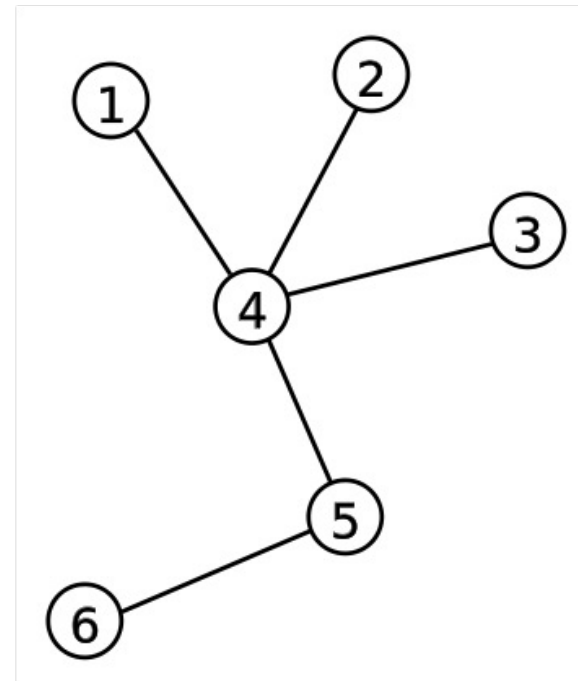
Uma árvore é um grafo:

- 1) Acíclico
- 2) Dois nós são acessíveis por um e só um caminho

Os cliques de árvores são só **pares de nós** adjacentes.

MRF baseado em árvores é tal que

$$Pr(\mathbf{X} = \mathbf{x}) = \prod_{\{i,j\} \in E} \phi(x_i, x_j)$$



MRF baseado em árvores

Fixado um conjunto de dados e uma MRF com suporte de uma árvore há uma e só uma distribuição que **maximiza a verosimilhança** dos dados:

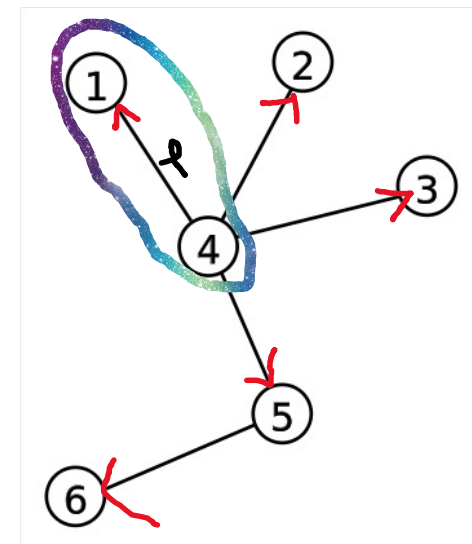
- 1) Escolhe-se um nó r ao acaso (ou fixa-se o primeiro) como raiz e uma aresta e com esse nó (à direita r é 4 e a aresta e é (4,1))
- 2) Esta escolha da raiz induz uma direção nas arestas (ver a vermelho) $i \rightarrow j$
- 3) Se $i \rightarrow j$ **não é** a aresta e então

$$\phi(x_i, x_j) = \frac{\text{Count}(T, (i, j), (x_i, x_j))}{\text{Count}(T, i, x_i)}$$

Se $i \rightarrow j$ **é** a aresta e então

$$\phi(x_i, x_j) = \frac{\text{Count}(T, (i, j), (x_i, x_j))}{m}$$

Onde $\text{Count}(T, (i, j), (x_i, x_j))$ é o número de vezes no dataset T que as variáveis i e j tomam **simultaneamente** os valores (x_i, x_j) e m é o tamanho do dataset T .



MRF em árvores

- 1) Quando se fixa uma árvore e um dataset T sabemos qual é o melhor MRF cuja distribuição é mais próxima da frequência dos dados.
- 2) Ao escolhermos uma árvore estamos a considerar um conjunto de independências (condicionais) entre as variáveis que evita o *overfitting*.
- 3) Falta saber como **escolher a melhor árvore**, para tal temos de entender o que é um grafo pesado!

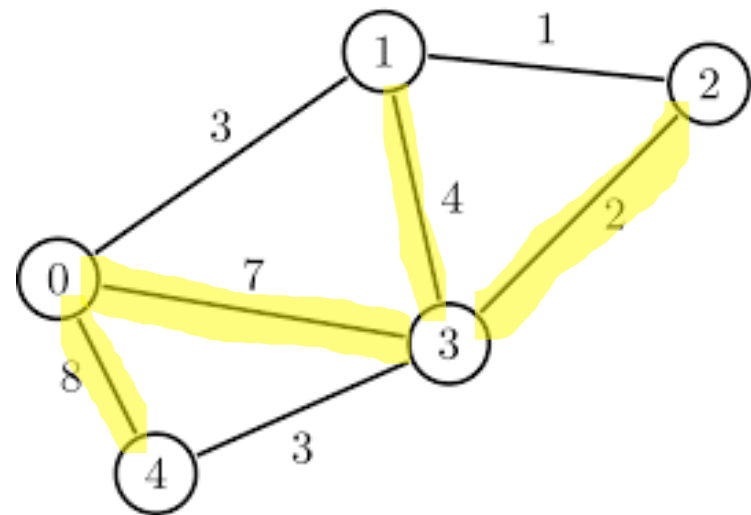
Grafos pesados

Um grafo pesado é um grafo que cada aresta é etiquetada com um peso, ou seja, $G=(N,E,w)$ onde

- 1) N é um conjunto de nós
- 2) E é um conjunto de arestas
- 3) $w:E \rightarrow \mathbb{R}$

Uma **árvore de extensão de peso maximal** (MST) é um subgrapho de G que é um árvore (com todos os nós) e cuja soma dos pesos das arestas é maximal entre todos os subgrafos que formam uma árvore.

Vamos aprender a calcular MST eficientemente nas aulas



Chow-Liu algorithm

- Input Data T de tamanho m para variáveis $\mathbf{X}=(X_1,\dots,X_n)$
 1. Construir o **grafo completo** e pesado G com \mathbf{X}
 2. Pesar a aresta entre i e j com a informação mútua de $I(i:j)$ com base na frequência de T, para todo i e j
 3. Retornar MST(G)

$$\text{onde } I(i:j) = \sum_{x_i \in D_i} \sum_{x_j \in D_j} \Pr(x_i, x_j) \log\left(\frac{\Pr(x_i, x_j)}{\Pr(x_i)\Pr(x_j)}\right) \quad (1)$$

$$\Pr(x_i, x_j) = \frac{\text{count}(T, (i, j), (x_i, x_j))}{m} \text{ e } \Pr(x_i) = \frac{\text{count}(T, i, x_i)}{m}$$

Nas fórmulas acima toma-se que $0 \cdot \log(0) = 0$

Classificador

E como classificar usando MRF's?

- 1) Particionam-se os dados originais T para cada valor possível da classe C

$T = \{T_0, \dots, T_{|D_C|-1}\}$ (no caso do cancro separam-se os dados nos caso com tumores benignos e malignos) cada T_c é chamado uma fibra de T

- 2) Aprende-se um MRFT M_c **usando cada fibra** T_c

- 3) $\Pr(\vec{V} = (x_1, \dots, x_n, c)) = \Pr(C = c) P_{M_c}(x_1, \dots, x_n)$

$$\text{onde } \Pr(C = c) = \frac{\text{count}(T, C, c)}{m}$$

Pseudo-contagens

Recorde que para cada MRF M_c

$$P_{M_c}(x_1, \dots, x_n) = \prod_{\{i,j\} \in E} \phi_{ij}(x_i, x_j)$$

Onde de acordo com o algoritmo do MRF para árvores

$$\phi_{ij}(x_i, x_j) = \frac{\text{Count}(T_c, (i,j), (x_i, x_j))}{\text{Count}(T_c, i, x_i)} \quad \text{ou}$$

$$\phi_{ij}(x_i, x_j) = \frac{\text{Count}(T_c, (i,j), (x_i, x_j))}{m_c} \quad \text{e } m_c \text{ é a dimensão de } T_c$$

No entanto (raramente) $\text{Count}(T_c, (i,j), (x_i, x_j))$ pode ser 0 e neste caso temos uma probabilidade nula (o que contradiz a def de MRF)

Assim, assume-se uma pseudo-contagem de $\delta = 0.2$ para todos os dados.

Pseudo-contagens

Ficando então a probabilidade

$$Pr_{M_c}(x_1, \dots, x_n) = \prod_{\{i,j\} \in E} \phi(x_i, x_j)$$

mas faz-se a seguinte alteração para que se tenham probabilidades positivas

$$\phi_{ij}(x_i, x_j) = \frac{\text{Count}(T_c, (i,j), (x_i, x_j)) + \delta}{\text{Count}(T_c, i, x_i) + \delta |D_j|} \quad (2) \quad \text{ou}$$

$$\phi_{ij}(x_i, x_j) = \frac{\text{Count}(T_c, (i,j), (x_i, x_j)) + \delta}{m_c + \delta |D_j| |D_i|} \quad (3)$$

e m_c é a dimensão de T_c (para a aresta especial)

Entrega

Classes

- Dataset
 - Count: recebe uma lista de variáveis e valores destas e retorna o número de vezes que estas variáveis tomam simultaneamente os esses valores no dataset.
 - Add: adiciona um vetor ao dataset.
 - Fiber: dado um valor da classe retorna a fibra (Dataset) associada a esse valor da classe.
- MRFT (Markov Random Field Tree)
 - Construtor que recebe uma árvore, e um dataset e coloca os $\phi_{ij}(x_i, x_j)$ em cada aresta da árvore (que podem ser vistos como uma matriz).
 - Prob: dado um vetor de dados (x_1, \dots, x_n) retorna a probabilidade destes dados no dataset.

Entrega

Classes

- Weighted Graph (não direcionado)
 - Add – recebe dois nós e uma peso e adiciona uma aresta entre os nós com este peso.
 - MST.
- Classifier
 - Construtor que recebe um *array* de MRFT's, um para cada valor da classe, e um *array* com a frequência das classes.
 - Classify: dados valores (x_1, \dots, x_n) das variáveis retorna o valor da classe mais provável.

Entrega

- Algoritmo de aprendizagem: Chow Liu (que será feita para cada fibra)
- Interface gráfica
 - Ler os dados, aprende o classificador e grava o dito (classificador) num ficheiro
 - Outra que lê o classificador e classifica observações/pacientes

Aplicação de aprendizagem

- Lê os dados e separa os dados em fibras (uma fibra para cada classe)
- Para cada fibra cria um MRF
 - Gera um grafo pesado completo cujas arestas são pesadas com a Informação Mútua - Eq. (1)
 - Aplica o algoritmo de Maximum Spanning Tree e cria uma árvore
 - Da árvore constroi um MRF usando Eq. (3) e (4)
- Os MRF's geram um array de MRF, que calcula a probabilidade de observar um dado com Eq. (2)
- Guarda o array de MRF's no disco

Aplicação de Classificação

- Lê o array de MRF's do disco
- Colocam-se valores de atributos com o objectivo de ser classificado
- Classifica de acordo com as probabilidade do array de MRF's.

Cotação final

- Dataset (1.5 val)
- MRFT e Classifier (2.0 val)
- Input/output de dados e resultados (2.5 val)
- Algoritmo de aprendizagem e inicialização (3 val)
- Aplicações gráficas (1 val)

Será entregue uma ficha de autoavaliação a preencher antes de cada oral.