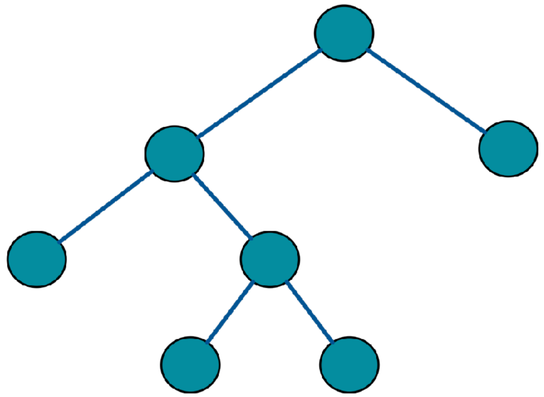
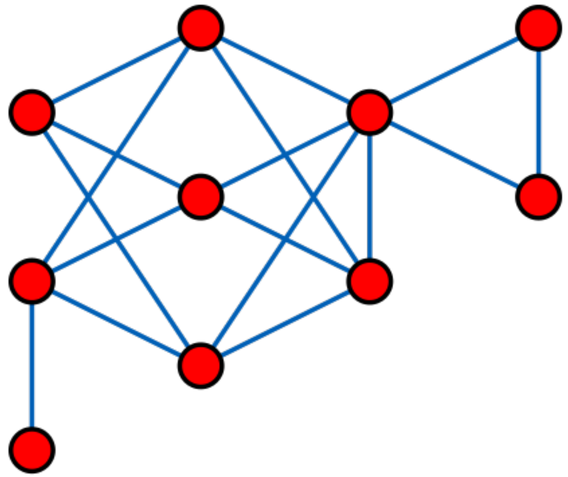
Projecto de
Algoritmos e Modelação Computacional

Francisco Dionísio, Paulo Mateus, João Ribeiro

2024/2025





Objectivo

- Implementar um algoritmo de aprendizagem baseado em *Campos de Markov Aleatórios (Markov Random Fields, MRFs)* — um bloco constituinte das Deep Belief Networks (quando se usam variáveis escondidas);

- Mais concretamente, implementar o algoritmo de Chow-Liu para aprender MRFs ótimas no contexto de classificação;

- Os dados biomédicos são públicos e fornecidos na página da disciplina, e provêm do UCI machine learning repository — http://archive.ics.uci.edu/ml/

A “história”

Classificadores

- Um classificador sobre um domínio D é uma função $f : D \to C$, onde C é um "conjunto de classes";

- Exemplo: Para a base de dados *Cancer,* o conjunto de classes é $C = \{\text{benign}, \text{malignant}\}$, e elementos de D correspondem a tuplos de medições.

- No nosso caso, teremos sempre $D = D_1 \times D_2 \times \cdots \times D_n$, onde n é o número de medições e D_i é o domínio da i-ésima medição.

Dados

- Um classificador é construído (“aprendido”) a partir de um conjunto de dados $T = \{T_1, T_2, \ldots, T_m\}$. Chamamos a m a *dimensão* dos dados;

- Os elementos de T são da forma $(d_1, d_2, \ldots, d_n, c)$, onde $(d_1, \ldots, d_n) \in D = D_1 \times \cdots \times D_n$ representam medições e $c \in C$ é a classe correspondente;

- No nosso caso, teremos sempre $D_i \subseteq \mathbb{N}$ (domínios discretizados) e $C \subseteq \mathbb{N}$, e portanto podemos ver o conjunto de dados T como uma matriz de dimensões $m \times (n + 1)$ com entradas naturais (a i-ésima linha corresponde a T_i).

Classificação e estimativas de distribuição

- Podemos interpretar medições $X_1, \ldots, X_n$ e a classe correspondente C como variáveis aleatórias cuja distribuição de probabilidades desconhecemos;

- Se conhecermos a distribuição de probabilidades de $(X_1, X_2, \ldots, X_n, C)$, podemos construir o classificador f usando uma *estimativa de máxima verosimilhança;*

- Mais precisamente, dadas medições $d_1, \ldots, d_n$, temos $f(d_1, \ldots, d_n) = c^*$, onde c^* é a classe "mais provável" para estas medições, que satisfaz
$$\Pr[(X_1, \ldots, X_n, C) = (d_1, \ldots, d_n, c^*)] \geq \Pr[(X_1, \ldots, X_n, C) = (d_1, \ldots, d_n, c)]$$
para qualquer classe $c \in C$.

Uma primeira tentativa

- Poderíamos tentar estimar a verdadeira distribuição de probabilidades de $(X_1, \ldots, X_n, C)$ aplicando a "lei dos grandes números" ao conjunto de dados T que temos disponível;

- **Lei dos grandes números:** Para estimar $\Pr[(X_1, \ldots, X_n, C) = (d_1, \ldots, d_n, c)]$, contar número de ocorrências de $(d_1, \ldots, d_n, c)$ em T, e dividir pela dimensão de T;

- Para evitar *overfitting*, esta estratégia requer um conjunto de dados extremamente grande… :(

Método alternativo — Markov Random Fields

Markov Random Field (MRF)

- Um conjunto de variáveis aleatórias $X = (X_1, \ldots, X_n)$ com $\Pr[(X_1, \ldots, X_n) = (x_1, \ldots, x_n)] > 0$ para todo o $(x_1, \ldots, x_n)$;
- Um grafo $G = (X, E)$ (o suporte do MRF);
- **Dependências limitadas:** Sejam A e B conjuntos disjuntos de vértices de G e $X_A = (X_i)_{i \in A}$. Então temos a decomposição

$$\Pr[X_A = x_A, X_B = x_B \,|\, X_S = x_S] = \Pr[X_A = x_A \,|\, X_S = x_S] \cdot \Pr[X_B = x_B \,|\, X_S = x_S]$$

para quaisquer x_A, x_B, x_S e qualquer conjunto S tal que qualquer caminho em G de A para B passa por S. Ou seja, X_A e X_B são independentes dado X_S.

Dificuldade de aprender MRFs

Depende do grafo de dependências!

- Quanto mais esparso o grafo, mais hipóteses de independência são feitas;
- Se o grafo é completo, voltamos à situação inicial.
- Só conhecemos um algoritmo de aprendizagem eficiente para **árvores!**

Algoritmo de Chow-Liu

O cerne do projecto

Classificar usando MRFs

Input: um conjunto de dados T (m vectores $(x_1, \ldots, x_n, c)$ onde $(x_1, \ldots, x_n)$ são medições e c a respectiva classe:

1. Para cada classe $c \in D_C$, definimos $T_c = \{(x_1, \ldots, x_n) : (x_1, \ldots, x_n, c) \in T\}$ como o conjunto das medições com classe associada c. Chamamos *fibra* a cada conjunto T_c.

2. Para cada $c \in D_C$ aprendemos uma MRF M_c usando T_c, através do algoritmo de Chow-Liu;

3. Definimos $\Pr[(X_1, \ldots, X_n, C) = (x_1, \ldots, x_n, c)] = p_C(c) \cdot p_{M_c}(x_1, \ldots, x_n)$, onde $p_C(c) = \frac{|T_c|}{m}$ e $p_{M_c}(x_1, \ldots, x_n)$ é a "probabilidade" atribuída às medições $(x_1, \ldots, x_n)$ pela MRF M_c.

4. **Classificação de $(x_1, \ldots, x_n)$:** A classe c^* que maximiza $p_C(c) \cdot p_{M_c}(x_1, \ldots, x_n)$.

Como encontrar a melhor árvore para um MRF?

Algoritmo de Chow-Liu: Input um conjunto T_c de m_c vectores de medições $(x_1, \ldots, x_n)$.

1. Construir o grafo **pesado** completo $G = (X, E)$ com pesos definidos abaixo;

2. A cada aresta $\{i, j\} \in E$ associar como peso a *informação mútua* entre X_i e X_j:

$$I(X_i; X_j) = \sum_{x_i \in D_i, x_j \in D_j} p_{i,j}(x_i, x_j) \cdot \log\left(\frac{p_{i,j}(x_i, x_j)}{p_i(x_i) \cdot p_j(x_j)}\right),$$

nº vezes que (X_i, X_j) tomam simultaneamente os valores (x_i, x_j) em T_c

onde $p_i(x_i) = \dfrac{Count(T_c, i, x_i)}{m_c}$, $p_{i,j}(x_i, x_j) = \dfrac{Count(T_c, (i,j), (x_i, x_j))}{m_c}$, e assumimos $0 \log 0 = 0$.

nº vezes que X_i toma o valor x_i em T_c

3. Retornar a *Maximum Weight Spanning Tree* de G.

   Intuição: É a árvore que retém mais "informação" entre as variáveis aleatórias.

MRFs baseados em árvores

Assumindo que já aprendemos um MRF M baseado numa árvore G, como obtemos a distribuição que maximiza a verosimilhança dos dados e que nos permite classificá-los?

Resumo: Obtemos uma orientação de G. Seja E o conjunto de arestas desta versão dirigida de G. A cada aresta $(i, j) \in E$ e possíveis medições (x_i, x_j) associamos um certo valor $\phi_{i,j}(x_i, x_j)$.

Usamos a estrutura da árvore dirigida $G = (X, E)$ para definir

$$p_M(x_1, \ldots, x_n) = \prod_{(i,j) \in E} \phi_{i,j}(x_i, x_j).$$

MRFs baseados em árvores

Como orientar a árvore e calcular os $\phi_{i,j}(x_i, x_j)$ para uma dada fibra T_c?

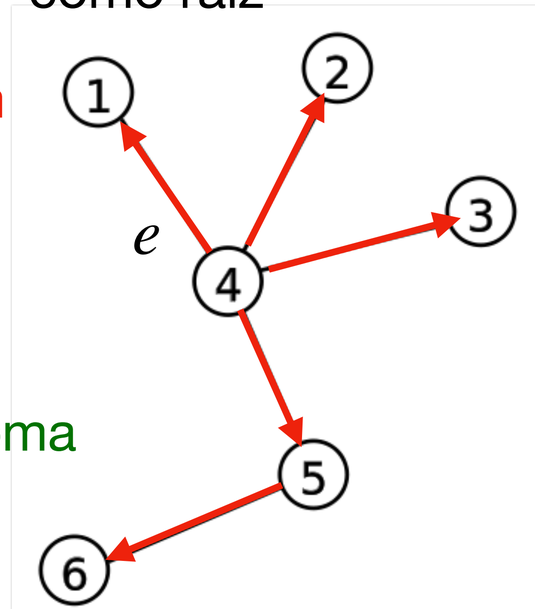
1. Escolhemos um vértice arbitrário como raíz (por exemplo, o primeiro) e uma aresta especial e adjacente a esse vértice;
2. A escolha da raíz induz uma orientação no grafo;
3. Se $(i,j) \neq e$, associamos $\phi_{i,j}(x_i, x_j) = \dfrac{Count(T_c, (i,j), (x_i, x_j))}{Count(T_c, i, x_i)}$
4. Se $(i,j) = e$, associamos $\phi_{i,j}(x_i, x_j) = \dfrac{Count(T_c, (i,j), (x_i, x_j))}{m_c}$

nº vezes que (X_i, X_j) tomam simultaneamente os valores (x_i, x_j) em T_c

nº vezes que X_i toma o valor x_i em T_c

m_c → cardinalidade de T_c

se escolhermos 4 como raíz



Na realidade, pseudo-contagens!

Por vezes, podemos ter $Count(T_c, (i,j), (x_i, x_j)) = 0$, o que contradiz a definição de MRF…

Assumimos então uma pseudo-contagem base $\delta = 0.2$ para todos os dados, e usamos uma expressão alternativa para $\phi_{i,j}(x_i, x_j)$ que garante sempre probabilidades positivas:

- Para $(i,j) \neq e$, definimos $\phi_{i,j}(x_i, x_j) = \dfrac{Count(T_c, (i,j), (x_i, x_j)) + \delta}{Count(T_c, i, x_i) + \delta |D_j|}$

- Para $(i,j) = e$, definimos $\phi_{i,j}(x_i, x_j) = \dfrac{Count(T_c, (i,j), (x_i, x_j)) + \delta}{m_c + \delta |D_j| |D_i|}$

Resumindo…

Se o MRF M_c aprendido para a fibra T_c é baseado na árvore dirigida $G_c = (X, E_c)$ com orientação induzida pela escolha da raíz r e aresta especial e adjacente a r, então definimos

$$p_{M_c}(x_1, \ldots, x_n) = \prod_{(i,j) \in E_c} \phi_{i,j}(x_i, x_j),$$

onde (com $\delta = 0.2$):

- $\phi_{i,j}(x_i, x_j) = \dfrac{Count(T_c, (i,j), (x_i, x_j)) + \delta}{Count(T_c, i, x_i) + \delta |D_j|}$ quando $(i,j) \neq e$;
- $\phi_{i,j}(x_i, x_j) = \dfrac{Count(T_c, (i,j), (x_i, x_j)) + \delta}{m_c + \delta |D_j| |D_i|}$ quando $(i,j) = e$, onde $m_c = |T_c|$ é o tamanho de T_c.

Entrega — 10 Janeiro 2025

Classes:

- Dataset:
  - Count: Recebe uma lista de índices de variáveis e valores destas, e devolve o número de vezes que estas variáveis tomam simultaneamente esses valores no dataset.
  - Add: Adiciona um vector ao dataset.
  - Fiber: Dado um valor da classe, devolve a fibra (Dataset) associada a esse valor da classe.

- MRFT (Markov Random Field Tree):
  - Construtor que recebe uma árvore e um dataset e calcula os valores $(\phi_{i,j}(x_i, x_j))_{x_i,x_j}$ (que podem ser representados por uma matriz indexada pelos possíveis valores de x_i e x_j) para cada aresta (i, j) da árvore.
  - Prob: Dado um vector de medições $(x_1, \ldots, x_n)$, devolve a probabilidade destes dados ocorrerem segundo a MRFT: $\text{Prob}(x_1, \ldots, x_n) = \prod_{(i,j)\in E} \phi_{i,j}(x_i, x_j)$.

Entrega — 10 Janeiro 2025

Classes (continuação):

- Weighted Graph:
  - Add — recebe dois nós e um peso e adiciona uma aresta entre os dois nós com esse peso;
  - Maximal weight spanning tree.
- Classifier:
  - Construtor recebe um array de MRFTs, um para cada valor da classe,e um array com a frequência das classes (os valores $p_C(c)$ para $c \in D_C$);
  - Classify — dados valores $(x_1, \ldots, x_n)$ das variáveis, devolve o valor da classe mais provável, ou seja, a classe c que maximiza $p_C(c) \cdot p_{M_c}(x_1, \ldots, x_n)$.
- Algoritmo de Chow-Liu (aplicado a cada fibra);
- Interface gráfica (minimalista é OK!):
  - Ler dados, aprender o classificador, gravá-lo num ficheiro;
  - Ler classificador e classificar pacientes.

Cotações

- Dataset (1.5 val)
- MRFT e Classifier (1.5 val)
- Input/output de dados e resultados (2 val)
- Algoritmo de aprendizagem e inicialização (3 val)
- Aplicações gráficas (1 val)
- Relatório minimalista (1 val)

Eficiência da implementação é parâmetro diferenciador na nota final!

Será entregue uma ficha de autoavaliação a preencher antes da discussão oral.