Projecto de Algoritmos e Modelação Computacional

2011/12 - 1ª parte

MEBiom, LMAC

Conteúdo

Capítulo 1

Objectivo

O objectivo do projecto é desenvolver um classificador baseado em redes de Bayes. O classificador é aprendido a partir de dados públicos que são fornecidos na página da disciplina, estes dados provêm do *UCI machine learning repository*.[1]

[1] `http://archive.ics.uci.edu/ml/`

Em particular, para avaliar a qualidade do classificador serão utilizadas as seguintes bases de dados:

- Cancer:
  `http://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagn`
  29

- Heart disease:
  `http://archive.ics.uci.edu/ml/datasets/Heart+Disease`

- Acute inflamation:
  `http://archive.ics.uci.edu/ml/datasets/Acute+Inflammations`

A qualidade do classificador será avaliada por intermédio de um método chamado *stratified cross validation*. Chama-se a atenção que, apesar de o exemplos a aplicar neste projecto se concentrarem em aplicações biomédicas, o domínio de aplicação do mesmo é muito mais extenso.

Capítulo 2

Conceitos básicos

2.1 Classificador

Um *classificador* sobre um domínio D é simplesmente um mapa $f : D \to C$ onde C é chamado o *conjunto de classes*. Por exemplo, para o caso da base da dados *Cancer*, o conjunto de classes é $C = \{\text{benign, malignant}\}$ 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 $d = (d_1, \ldots, d_n)$.

2.2 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 $T = \{T_1, \ldots, T_m\}$ e $T_j = (d_{1j}, \ldots, d_{nj}, c_j)$ 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.

2.3 Classificar vs estimar

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

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

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

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

2.4 Redes de Bayes

Para ultrapassar a limitação de não se possuir dados suficientemente grandes, supõe-se que existem dependências directas entre as variáveis e que estas dependências estão descritas num grafo acíclico $G = (\mathcal{X}, E)$ onde $\mathcal{X} = \{X_1, \ldots X_n, Y\}$ tal que $(Y, X_i) \in E$ para $1 \leq i \leq n$. O facto de todas as variáveis X_i dependerem de Y prende-se com o facto de que, em princípio, X_i não é independente de Y, pois caso contrário X_i não serve para classificar (ou estimar) Y. Assim podemos decompor a distribuição de probabilidade do vector $\vec{V}$ da seguinte forma

$$\Pr(\vec{V} = (d_1, \ldots, d_n, c)) = \Pr(Y = c) \prod_{i=1}^{n} \Pr(X_i = d_i | \vec{P}_i = (d_{i,1} \ldots d_{i,k_i}, c)) \qquad (2.4.1)$$

onde $\vec{P}_i = (X_{i,1}, \ldots, X_{i,k_i}, Y)$ é um vector constituido pelos pais de X_i no grafo G.

Supondo que, com cada nó X_i tem um pai X_{P_i} para além de Y, com a excepção de um nó X_r que só tem Y como pai, podemos rescrever a Equação (2.4.1) da seguinte forma

$$\Pr(\vec{V} = (d_1, \ldots, d_n, c)) = \Pr(Y = c)\Pr(X_r = d_r | Y = c) \prod_{i \neq r} \Pr(X_i = d_i | X_{P_i} = d_{P_i}, Y = c). \quad (2.4.2)$$

Assim para obter a distribuição de $\vec{V}$ basta conhecer as distribuições Y, $X_r|_Y$ e $X_i|_{(X_{P_i},Y)}$. Note que como todo os dados estão discretizados, e $|D_i|$ é finito, as variáveis Y, $X_r|_Y$ e $X_i|_{(X_{P_i},Y)}$ são variáveis multinomiais. Neste caso já se torna possível estimar as distribuições Y, $X_r|_Y$ e $X_i|_{(X_{P_i},Y)}$, utilizando a lei dos grande números, mesmo com dados relativamente pequenos.

Com generalidade, uma rede de Bayes é um tuplo (G, Θ) onde $\Theta = \{\Theta_{i|w_i}\}_{i \in N, w_i \in D_{\vec{P}_i}}$ e $\Theta_{i|w_i}$ é uma distribuição multinomial para a variável X_i e $D_{\vec{P}_i}$ é o domínio dos pais de X_i em G. Fixado um grafo G as distribuições multinomiais em Θ que maximizam a verosimilhança dos dados T são dadas por

$$\Theta_{i|w_i}(d_i) = \frac{|T_{d_i,w_i}|}{|T_{w_i}|}$$

onde T_{d_i,w_i} é o conjunto de amostras de T onde a variável X_i toma o valor d_i e os seus pais tomam o valor w_i e, de forma semelhante T_{w_i} é o conjunto de amostras de T onde os pais de X_i tomam o valor w_i. Caso T_{w_i} seja vazio, $\Theta_{i|w_i}$ deverá ser uniforme. Esta distribuição é chamada a distribuição das frequências observadas (DFO).

2.5 Aprendizagem de Redes de Bayes

Pelo o que foi apresentado anteriormente, para aprender redes Bayes dado T basta aprender o grafo orientado G já que Θ é obtido das DFO's. Encontrar o grafo que maximiza a verosimilhança de T é um problema NP-completo e para o qual não se espera haver solução eficiente. Mais, ao maximizar a verosimilhança obtêm-se grafos completos e não grafos esparsos. Mas mais uma vez, para grafos completos as DFO's associadas a dados pequenos fazem overfitting. A solução é restringir a aprendizagem a grafos com estruturas mais simples, e no caso deste projecto só serão aprendidas árvores, isto é, o grafo G restringido às variáveis $X_1, \ldots X_n$ forma uma árvore.

Como derivado nas aula teórica T7, a árvore A que maximiza a verosimilhança de T é a expansão arborescente mais pesada do grafo completo com nós $X_1, \ldots X_n$ e onde cada aresta de X_i para X_j é pesada por

$$I_T(X_i, X_j|C).$$

Note que $I_T(X_i, X_j|C)$ é a informação mútua condicional de X_i com X_j dado C medida com a distribuição de probabilidade obtida pela DFO. Esta expansão pode ser obtida em tempo polinomial, pelo algoritmo *greedy* MST, discutido na aula teórica T8.

Capítulo 3

Tipos de dados

Os tipos de dados a serem utilizados neste projecto são os seguintes:

3.1 Amostra

- `emptyS`: retorna a amostra vazia;
- `add`: recebe um vector e um amostra e acrescenta o vector à amostra;
- `length`: recebe uma amostra e retorna o comprimento da amostra;
- `element`: recebe uma amostra e uma posição e retorna o vector da amostra;
- `count`: recebe uma amostra, um vector de variáveis e um vector de valores e retorna o número de ocorrências desses valores para essas variáves na amostra;
- `join`: recebe duas amostra e retorna uma nova amostra com as duas concatenadas;

3.2 Grafos orientados

- `disc`: recebe um natural n e retorna o grafo com n nós e sem arestas.
- `add_edge`: recebe um grafo e dois nós e adiciona ao grafo uma aresta de um nó para outro.
- `parents`: recebe um grafo e um nó e retorna a lista de nós que são pais do nó.

3.3 Grafos com pesos

- `discWD`: recebe um natural n e retorna o grafo com pesos com n.
- `add_Wedge`: Recebe um grafo, dois nós e um peso real e adiciona o grafo uma aresta de um nó para outro com o peso do real.
- `MST`: Recebe um grafo pesado e retorna a expansão arborescente maximal como um grafo orientado.

3.4 Redes Bayesianas

- `newBN`: Recebe um grafo e um conjunto de dados e retorna a rede de Bayes com as distribuições DSO.
- `prob`: Recebe uma rede de Bayes e um vector e retorna a probabilidade desse vector.