Projecto de Algoritmos e Modelação Computacional

Agrupamento (*clustering*) para modelos farmacocinéticos

Optimização com *Simulated annealing*

2016/17

MEBiom e LMAC

Conteúdo

Capítulo 1

Objectivo

A farmacocinética estuda a concentração de fármacos ao longo do tempo no corpo humano. Nem todos os pacientes apresentam as mesmas concentrações de um fármaco, mesmo quando administrados com a mesma dose. Há, no entanto, padrões de resposta importantes a analisar com vista ao desenvolvimento de terapias personalizadas. Este projecto visa encontrar classes de resposta a fármacos para estratificar os pacientes nas mesmas.

Concretamente, o objetivo do projeto é desenvolver em Java um classificador não supervisionado para curvas que descrevem modelos farmacocinéticos, baseado no algoritmo de *Expectation Maximization* para Gaussianas.

O classificador é aprendido a partir de dados públicos fornecidos em *monolix - http://accp1.org/pharmacometrics/cssolutionmonolix.htm* que serão colocados numa base de dados MySQL. O resultado final corresponderá a uma lista de **classes de comportamento** do organismo na absorção de um certo fármaco numa dada população, em que cada classe é descrita por um vector de parâmetros.

Capítulo 2

Conceitos básicos

2.1 Modelos farmacocinéticos baseados em compartimentos

No projecto em questão será considerado apenas modelos com **um compartimento**. Este modelos servem para estimar a concentração de fármacos no organismo. No modelo com um compartimento em questão considera-se que a droga é administrada oralmente, e que cada pessoa tem uma capacidade k_a de absorção do fármaco e uma capacidade k_e de eliminar o fármaco. Estas capacidade dão origem às seguinte equações diferenciais para a *consumo de fármaco* $I(t)$ e quantidade deste no organismo $Q(t)$ ao longo do tempo:

$$\begin{aligned} I'(t) &= -k_a I(t) \\ Q'(t) &= -k_e Q(t) + k_a I(t) \end{aligned} \tag{2.1.1}$$

restritas às seguintes condições iniciais

$$I(0) = \text{Dose} \times F \text{ e } Q(0) = 0$$

onde $F \in [0, 1]$ é uma constante que indica a taxa de fármaco que o organismo tem acesso, neste projecto consideramos $F = 0.5$.

As curvas dizem respeito à concentração $C(t)$ do fármaco no sangue (grandeza que pode ser medida na prática), sendo claro que

$$C(t) = \frac{Q(t)}{V}$$

onde V é o volume do organismo. Na prática, os fármacos são administrados assumindo que k_a e k_e são iguais em toda a população, sendo apenas distinguido o V. No entanto, verifica-se que existe grande variabilidade nestas constantes que justifica dar diferentes doses a indivíduos com o mesmo volume (medicina personalizada).

Da solução para a equação (2.1.1) obtemos:

$$C(t) = \frac{(\text{Dose} \times F)}{V} \frac{k_a}{k_a - k_e}(e^{-k_e t} - e^{-k_a t})$$

e portanto, assume-se que a curva $C(t)$ é da forma

$$C(t) = a_1 e^{-b_1 t} + a_2 e^{-b_2 t}$$

pretendendo-se estimar os parâmetros a_i e $b_i = k_i$, assumindo ainda que

$$a_1 = -a_2 > 0, \tag{2.1.2}$$

$$0 < b_1 < b_2 < 5, \tag{2.1.3}$$

Onde 5 é um limite obtido tendo em conta as unidades utilizadas.

2.2 Algoritmo EM para misturas de Gaussianas

Como iremos ver, o algoritmo de agrupamento baseia-se no algoritmo de EM para misturas de Gaussianas (no caso em questão são unidimensionais) dado que vamos assumir que o erro na medição da grandeza de concentração tem uma distribuição normal em cada instante do tempo.

No caso de misturas de Gaussianas, os dados são um conjunto de pontos $\{x_i\}_{i=1,\dots K}$ com $x_i \in \mathbb{R}$ i.i.d. de uma distribuição desconhecida, correspondente a uma mistura de M Gaussianas $p_\theta(x) = \sum_{j=1}^{M} w_j g_j(x)$ onde cada Gaussiana tem como parâmetros μ_j, σ_j^2 e os pesos w_j estão normalizados tal que $\sum_j w_j = 1$. O conjunto θ inclui todos estes parâmetros. M corresponderá ao número de classes a serem encontradas. Neste projecto consideramos que o M é fixo, no entanto, poderá ser relevante na prática considerar um M variável de acordo com os dados da população.

O objectivo é encontrar os parâmetros que maximizam a verosimilhança dos dados, isto é, encontrar θ tal que $\arg_\theta \max p_\theta(x_1 \dots x_K)$ onde

$$p_\theta(x_1 \dots x_K) = \prod_{k=1}^{K} p_\theta(x_k)$$

o que é equivalente a maximizar o logaritmo da verosimilhança

$$\log(p_\theta(x_1 \dots x_K)) = \sum_{k=1}^{K} \log(p_\theta(x_K))$$

A ideia do algoritmo de EM consiste em calcular uma sequência de parâmetros

$$\theta^0 \dots \theta^n$$

partindo de um conjunto de parâmetros inicial θ^0 de tal forma que

$$p_{\theta^m}(x_1 \dots x_K) < p_{\theta^{m+1}}(x_1 \dots x_K)$$

Para este fim, considera-se uma família de variáveis escondida $\{Y_k\}_{k=1,\dots K}$ que toma valores no conjunto $\{1, \dots, M\}$ tal que

$$P(Y_k = j|D)$$

é a probabilidade de i-ésima amostra x_i ter sido amostrado de acordo com a j-ésima Gaussiana dados os dados D observados

Considera-se de seguida o valor esperado de acordo com a distribuição $p_{\theta^m}(Y|D)$ para $\log(p_\theta(x_1 \dots x_K))$

$$\begin{aligned} \log(p_\theta(X)) &= E[\log(p_\theta(X))] \\ &= E[\log(p_\theta(X,Y))] - E[\log(p_\theta(Y|X))] \end{aligned}$$

Denota-se por $Q(\theta, \theta^m)$ o valor $E[\log p_\theta(X,Y)]$.

Lema: Seja θ tal que $Q(\theta, \theta^m) > Q(\theta^m, \theta^m)$, então $\log(p_\theta(X)) > \log(p_{\theta^m}(X))$.

Assim, após calcular $Q(\theta, \theta^m)$ (E-step) basta encontrar $\arg_\theta \max Q(\theta, \theta^m)$ (M-step). Tal é possível ser feito analiticamente para misturas (ver cálculos no quadro da aula). O Algoritmo de EM consistem em aplicar estes passos sucessivamente até ser obtido um ponto fixo (a menos de $\varepsilon > 0$).

2.3 Aplicação à farmacocinética

No caso de dados farmacocinéticos temos que a amostra

$$y_i(t) = C(t) + \varepsilon$$

onde $\varepsilon \sim N(0, \sigma^2)$ e logo

$$p(y_1(t_1), \dots, y_1(t_n), \dots y_K(t_n)) = \sum_{j=1}^{M} w_j \prod_{i=1}^{K} \prod_{\ell=1}^{n} g_j(y_i(t_\ell), t_\ell)$$

onde $g_j(y,t) \sim N(C_j(t), \sigma_j^2)$.

Objectivo: Encontrar $\theta = \{\theta_j\}_{j=1\dots M}$ e $\theta_j = \{w_j, \sigma_j, a_{1j}, a_{2j}, b_{1j}, b_{2j}\}$ que maximizam a verosimilhança dos dados, restrito a que $a_{1j} = -a_{2j}$ e que $b_{1j} < b_{2j}$ para que $C(t)$ seja positiva.

Recorde que a função f que queremos estimar é

$$f(\theta_j, t) = \sum_{i=1}^{2} a_{ij} e^{-b_{ij} t} \tag{2.3.4}$$

que representa a concentração da droga no organismo do componente j no instante t. Utilizando a técnica semelhante ao EM para misturas Gaussianas, temos que:

$$Q(\theta, \theta^{(k)}) = \sum_{j=1}^{M} \sum_{i=1}^{K} X_{ij} \log w_j^{(k)} p(y_i|\theta_j^{(k)}) \tag{2.3.5}$$

com

$$p(y_i|\theta_j) = \frac{1}{(2\pi\sigma_j^2)^{\frac{n}{2}}} e^{\frac{-1}{2\sigma^2} \sum_{\ell=1}^{n} (y_i(t_\ell) - f(\theta_j, t_\ell))^2} \tag{2.3.6}$$

e

$$X_{ij} = \frac{w_j p(y_i|\theta_j)}{\sum_{u=1}^{M} w_u p(y_i|\theta_u)}. \tag{2.3.7}$$

Observe que tanto o denominador como o numerados são bastantes baixos, e no Java pode acontecer que ambos tomem o valor 0. Para que tal não aconteça (nos dados oferecidos) devem multiplicar o numerador e o denominador por e^{500}.

Seguindo o algoritmo de EM para Gaussianas, adaptando apenas o valor médio obtemos que a alteração dos w_i em cada iterada do algoritmo que maximiza $Q(\theta, \theta^{(k)})$ é feita de acordo com o seguinte equação:

$$w_j^{(k+1)} = \frac{1}{K} \sum_{i=1}^{K} X_{ij}^{(k)}. \tag{2.3.8}$$

Para encontrar o valor de σ_j^2 que maximiza $Q(\theta, \theta^{(k)})$ é necessário derivar $Q(\theta, \theta^{(k)})$ a σ^2 e encontrar um zero para o qual a segunda derivada seja negativa. Assim, temos que

$$\frac{\partial Q(\theta, \theta^{(k)})}{\partial \sigma_j^{2(k)}} = \sum_{i=1}^{K} X_{ij} \left(-\frac{K}{2\sigma_j^{2(k)}} + \frac{1}{2\left(\sigma_j^{2(k)}\right)^2} \sum_{l=1}^{n} (y_i(t_l) - f(\theta_j^{(k)}, t_l))^2 \right)$$

Logo $\frac{\partial Q(\theta, \theta^{(k)})}{\partial \sigma_j^{2(k)}} = 0$ sse

$$\sigma_j^{2(k+1)} = \frac{\sum_{i=1}^{K} \sum_{l=1}^{n} X_{ij} (y_i(t_l) - f(\theta_j^{(k+1)}, t_l))^2}{\sum_{i=1}^{K} n X_{ij}} \tag{2.3.9}$$

Note que vai ter de actualizar todos os parâmetros de θ antes de actualizar σ e que $f(\theta_j^{(k+1)}, t_l)$ depende apenas de $a_j^{(k+1)}, b_{1j}^{(k+1)}, b_{2j}^{(k+1)}$.

Para simplificar a notação vamos utilizar y_{il} em vez de $y_i(t_l)$. É relativamente fácil verificar que a segunda derivada é negativa neste ponto.

Para ser possível actualizar o θ é necessário derivar $Q(\theta, \theta^{(k)})$ em ordem a $a_j = a_{1j} = -a_{2j}$ para encontrar o máximo e assim iterar o valor de a_j. Neste caso, $\frac{\partial Q(\theta,\theta^{(k)})}{\partial a_j^{(k)}} = 0$ sse

$$a_j^{(k+1)} = \frac{\sum_{i=1}^{K}\sum_{l=1}^{n} X_{ij} y_{il}(e^{-b_{1j}^{(k)} t_l} - e^{-b_{2j}^{(k)} t_l})}{\sum_{i=1}^{K}\sum_{l=1}^{n} X_{ij}\left(e^{-b_{1j}^{(k)} t_l} - e^{-b_{2j}^{(k)} t_l}\right)^2} \tag{2.3.10}$$

e mais uma vez, verifica-se que este ponto é um máximo. O caso mais complicado ocorre quando se tenta maximizar b_{1j} e b_{2j}.

Neste projecto vamos adoptar um técnica de *simulated annealing* (arrefecimento simulado) para optimizar os parâmetros b_{1j} e b_{2j} que se descreve de seguida.

2.3.1 Arrefecimento simulado

Uma técnica simples para optimizar os parâmetros b_{1j} e b_{2j} da função (2.3.5) é designada por arrefecimento simulado. A ideia é começar com uma estimativa inicial aleatória θ que contém os w's, σ's e a's calculados de acordo com (2.3.8), (2.3.9) e (2.3.10) e os b_{1j} e b_{2j} são escolhidos de forma aleatória (mas sempre nas restrições expressas na equação (2.1.3))

O processo de arrefecimento consiste num ciclo `while` para cada $j \in 1 \ldots M$. Em cada iterada deste ciclo, fixado um j, e encontram-se vizinhos aleatórios b'_{1j} e b'_{2j} de b_{1j} e b_{2j}, Testa-se de seguida se $Q(\theta', \theta^{(k)}) > Q(\theta, \theta^{(k)})$ (na realidade pode-se testar diretamente se $\log p_{\theta'}(D) > \log p_\theta(D)$ usando a função prob). Caso tal se verifique passamos $\theta = \theta'$. Caso não se verifique aceita-se θ como ótimo com uma probabilidade p baixa (um parâmetro de optimização que no caso concreto será 0.0001). A este processo chama-se arrefecimento. Se assim o desejar poderá primeiro optimizar o b_{1j} e só depois o b_{2j}.

Depois de encontrado um máximo, volta-se a aquecer o sistema, isto é escolhe-se outro b_{1j} e b_{2j} de forma aleatória e volta-se a arrefecer o sistema até ser encontra-do outro máximo. Entre cada arrefecimento, vai-se optando sempre pelo máximo que foi encontrado até ao momento. Um dos aquecimentos deverá conter como b_{1j} e b_{2j} os valores anteriores, isto é, $b_{1j}^{(k)}$ e $b_{2j}^{(k)}$.

Termina-se após R aquecimentos (em que R é um parâmetro do sistema, que no nosso caso será 10000) escolhendo como máximo o valores de $b_{1j}^{(k)}$ e $b_{2j}^{(k)}$ que optimizaram (2.3.5) entre todos os arrefecimentos.

2.3.2 Condições de paragem

Falta indicar quais as condições de paragem para o simulated annealing (SA) e para o EM em geral.

A convergência do SA é feita da seguinte forma. Os valores das iteradas de b_1 e b_2 têm de verificar a seguintes condição:

- $0 < b_{1j}^{(k+1)} < b_{2j}^{(k)}$ – condição para $b_{1j}^{(k+1)}$;
- $b_{1j}^{(k+1)} < b_{2j}^{(k+1)} < 5$ – condição para $b_{2j}^{(k+1)}$ on o valor 5 é uma heurística.

Se os valores das iteradas verificarem sempre a condição respectiva, então o método deve terminar depois de 10000 (dez mil) iteradas.

Se por outro lado assim que uma iterada de $b_{dj}^{(k+1)}$ ficar fora do intervalo respectivo, neste caso existe um cotovelo (knee choice - apesar de "joelho" ser diferente de "cotovelo") e é necessário proceder a uma escolha diferente do $b_{dj}^{(k+1)}$ (para os dados fornecido este problema apenas acontece para o b_{2j}). A escolha é feita da seguinte forma

1. Reinicia-se o método de SA (no máximo com 10000 iteradas)
2. Termina-se o ciclo do SA se os limites b_{dj} são violados;
3. O aluno pode reiniciar outro aquecimento ou implementar uma heurística tal como:
   - Caso os limites sejam violados decrementa-se $b_{2j} = b_{2j} - 0.2$ e voltamos para o passo 1.
   - Caso o b_{dj} fique menor que 0 então $b_{dj}^{(k+1)} = b_{dj}^{(k)}$.

Quanto à convergência do EM, este termina quando

$$(b_{dj}^{(k+1)} - b_{dj}^{(k)})^2 < 0.000001$$

para $d = 1$ e $d = 2$. Não se faz nenhuma consideração sobre a, pois a curva é muito mais sensível às variáveis b's.

Capítulo 3

Classes para a 1ª entrega

A entregar a 21 de Abril de 2017. As classes a serem utilizadas neste projeto são as seguintes:

3.1 Amostra

- `add`: recebe um vector com três campos (índice, tempo e valor) e acrescenta o vector à amostra;
- `length`: retorna o comprimento da amostra;
- `element`: recebe uma posição e retorna o vector da amostra;
- `indice`: retorna uma lista de pares (tempos, valor) para um dado indice;
- `join`: recebe uma amostra e retorna uma nova amostra com as duas concatenadas;

3.2 Misturas de Gaussianas

- `mix`: Método construtor que recebe um inteiro (número de misturas), uma lista de parâmetros θ;
- `prob`: Recebe uma lista de pontos ao longo do tempos retorna a probabilidade dessa lista de pontos ser observada pela mistura;
- `theta`: Retorna a lista de parâmetros actual;
- `update`: Método que recebe uma lista de parâmetros θ e actualiza a mesma.

O tipo lista de parâmetros pode ser implementada usando as classes `Collection` do Java.

Capítulo 4

2ª entrega

Na segunda entrega deverá ser implementada a aplicação gráfica, pela qual deve ser possível:

- Ler os dados de uma base de dados.
- Associar um ficheiro *.theta com a aproximação inicial de θ.
- Fazer a aprendizagem não supervisionada, tal como descrito na Secção 2, dos parâmetros.
- Apresentar quais foram os parâmetros aprendidos por intermédio de um ficheiro de output.

Deverá ser elaborado um relatório em que conste:

- Documentar as opções tomadas no projecto bem como a justificação de alterações à 1a entrega.
- Pequeno manual de utilização.
- Experimentação dos dados fornecidos onde se deve ilustrar as curvas aprendidas (usando e.g. Mathematica)

Capítulo 5

Cotação

São avaliadas as opções de velocidade, bem como a documentação do código.

- Tipos de dados/Classes (3 val)
  - Amostra (1.5 val)
  - Mistura de Gaussianas (1.5 val)
- Algoritmo de Aprendizagem (3 val)
- Input/output de dados/resultados (2 val)
- Aplicação Gráfica (1 val)
- Relatório (1 val)