Clustering cancer data with Bayesian nonparametrics and F#

Evelina Gabasova

MRC Biostatistics Unit
University of Cambridge
Precision medicine
Precision medicine
Integrative clustering

Breast cancer

Gene expression

microRNA expression

Reverse-phase protein array

Methylation data
Integrative clustering

Breast cancer

Data

Data

Data

Data
Bayesian nonparametrics

Parametric models

\[ P(x \mid \theta, D) = P(x \mid \theta) \]

Nonparametric models

Models are potentially infinite

Complexity adapts to data
Dirichlet process mixture

- Distribution over distributions
- Limit of a finite mixture model

\[ p(x) = \sum_{k=1}^{K} \pi_k p(x \mid \theta_k) \]
Chinese restaurant process
Chinese restaurant process

sampling from a Dirichlet process mixture model

$\theta_1$
Chinese restaurant process

sampling from a Dirichlet process mixture model
Back to Dirichlet processes

Parameters for each customer are drawn from a Dirichlet process

\[ G \sim \text{DP}(\alpha, H) \]

\( \alpha \) concentration parameter
\( H \) base measure

\[ G = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k} \]
Integrative clustering

Parameters for each customer are drawn from a Dirichlet process

We can expand $\theta_k$ into tuples of parameters, one for each data type
Integrative clustering

Parameters for each customer are drawn from a Dirichlet process

\[ \theta^1_k, \theta^2_k, \theta^3_k, \theta^4_k \]

We can expand \( \theta_k \) into tuples of parameters, one for each data type

Breast cancer

Gene expression

microRNA expression

Reverse-phase protein array

Methylation data
Results
Results
Scientific programming

Scripting languages

fast prototyping, easy to use

R, Python, Matlab
Functional programming

\[ x = x + 1 \]
Legibility

\[ X + \log \left( \sum_{i=1}^{N} \exp \{ x_i - X \} \right) \]

where \( X = \max \{ x_i; i = 1, \ldots, N \} \)

```ml
let logSumExp xs =  
let maxValue = Array.max xs  
xs  |> Array.map (fun x -> exp(x - maxValue))  
 |> Array.sum  
|> log  
|> (+) maxValue
```
Legibility

\[ X + \log \left( \sum_{i=1}^{N} \exp \{ x_i - X \} \right) \]

where \( X = \max \{ x_i; i = 1, \ldots, N \} \)

```ocaml
let logSumExp xs = 
let maxValue = Array.max xs xs 
|> Array.map (fun x -> exp(x - maxValue)) 
|> Array.sum 
|> log 
|> (+) maxValue
```
Legibility

\[ X + \log \left( \sum_{i=1}^{N} \exp \{ x_i - X \} \right) \]

where \( X = \max \{ x_i; i = 1, \ldots, N \} \)

```haskell
let logSumExp xs =
    let maxValue = Array.max xs
    xs
    |> Array.map (fun x -> exp(x - maxValue))
    |> Array.sum
    |> log
    |> (+) maxValue
```
Why F#?
Why F#?

Why F#?

let newas =
data.Contexts
    |> Array.map (fun context ->
    let pis = state.ContextWeights.[context]
    let rawPriorPis = Array.create pis.Length 1.0
    let value = sampleDirichletConcentration (hyperprior.AlphasPrior.[context])
    (context - 1))

|> toFixedValue
{ state = state, state = newas}
    FixedValue(state, state)
    // do not
    state

// Context-specific
// **************

// Sample from conditional Dirichlet distribution with random walk Metropolis-Hastings
let sampleDirichlet_MetropolisHastings (currentValues: float[]) =
priorConcentration loglikFunction =
    // 1. Add random walk proposal to current values
let randomWalkProposal = Normal(0.0, 1.0, rnd)
let proposal_unnorm =
currentValues
    |> Array.map (fun x -> x + 0.1 * randomWalkProposal.Sample())
Case study

Rewrite an algorithm from Matlab into F#

adaptive rejection sampling
Case study

% value of the lower bound
if x<min([lowerHull.left])
    lhVal = -inf;
elseif x>max([lowerHull.right]);
    lhVal = -inf;
else
    for li=1:length(lowerHull)
        left = lowerHull(li).left;
        right = lowerHull(li).right;
        if x>=left && x<=right
            lhVal = lowerHull(li).m*x + lowerHull(li).b;
            break;
        end
    end
end

MatLab
Pattern matching

% value of the lower bound
if \( x < \text{min}([\text{lowerHull.left}]) \)
    \( \text{lhVal} = -\infty \);
elseif \( x > \text{max}([\text{lowerHull.right}]) \);
    \( \text{lhVal} = -\infty \);
else
    for \( \text{li} = 1 : \text{length(lowerHull)} \)
        \( \text{left} = \text{lowerHull(li).left} \);
        \( \text{right} = \text{lowerHull(li).right} \);
        if \( x \geq \text{left} \) \&\& \( x \leq \text{right} \)
            \( \text{lhVal} = \text{lowerHull(li).m} \times x + \text{lowerHull(li).b} \);
            break;
        end
    end
end

// value of the lower bound
let \( \text{lhVal} = \)
match (lowerHull, x) with
| \text{OutsideOnLeft} \to -\infty
| \text{OutsideOnRight} \to -\infty
| \text{InsideInterval lh} \to \text{lh.M} \times x + \text{lh.B}

F#
Pattern matching

% value of the lower bound
if x < min([lowerHull.left])
    lhVal = -inf;
elseif x > max([lowerHull.right])
    lhVal = -inf;
else
    for li = 1:length(lowerHull)
        left = lowerHull(li).left;
        right = lowerHull(li).right;
        if x >= left && x <= right
            lhVal = lowerHull(li).m*x + lowerHull(li).b;
            break;
        end
    end
end

// value of the lower bound
let lhVal =
    match (lowerHull, x) with
    | OutsideOnLeft -> -infinity
    | OutsideOnRight -> -infinity
    | InsideInterval lh -> lh.M * x + lh.B

F#
Dealing with real-world data

Andréj Karpathy
@karpathy

Students in my class: Project proposal time: "We will solve computer vision" Project Milestone time: "We got stuck on preprocessing data"

6:13 AM - 17 Feb 2015

RETWEETS 24  FAVORITES 65
Type providers

F# Data - typed access to external data sources

Csv files

JSON, XML, HTML, SQL
RProvider
Summary

• Bayesian nonparametrics are nice flexible models
• Complexity of the model adapts to data
• There are efficient inference algorithms

• Functional programming is great for maths
• F# has great tools for accessing external sources
Thank you

@evelgab

evelinag.com

evlna@evelinag.com