Bayesian Inference of First Order Markov Chains

Sai Bhargav Yalamanchi, Giorgio Alfredo Spedicato

2015-08-03

The *markovchain* package provides functionality for maximum a posteriori (MAP) estimation of the chain parameters[1] by Bayesian inference. It also computes the probability of observing a new data set, given a (different) data set. This vignette provides the mathematical description for the methods employed by the package.

Notation and set-up

The data is denoted by D, the model parameters (transition matrix) by θ. The object of interest is $P(\theta|D)$ (posterior density). $\mathcal{A}$ represents an alphabet class, each of whose members represent a state of the chain. Therefore

$$D = s_0 s_1 ... s_{N-1}, s_t \in \mathcal{A}$$

where N is the length of the data set. Also,

$$\theta = \{p(s|u), s \in \mathcal{A}, u \in \mathcal{A}\}$$

where $\sum_{s \in \mathcal{A}} p(s|u) = 1$ for each $u \in \mathcal{A}$.

Our objective is to find θ which maximises the posterior. That is, if our solution is denoted by $\hat{\theta}$, then

$$\hat{\theta} = \underset{\theta}{argmax} P(\theta|D)$$

where the search space is the set of right stochastic matrices of dimension $|\mathcal{A}|x|\mathcal{A}|$.

$n(u, s)$ denotes the number of times the word us occurs in D and $n(u) = \sum_{s \in \mathcal{A}} n(u, s)$. The hyperparameters are similarly denoted by $\alpha(u, s)$ and $\alpha(u)$ respectively.

Method

Given D, its **likelihood** is given by

$$P(D|\theta) = \prod_{s \in \mathcal{A}} \prod_{u \in \mathcal{A}} p(s|u)^{n(u,s)}$$

Conjugate priors are used to model the **prior** $P(\theta)$. The reasons are two fold

- Exact expressions can be derived for the MAP estimates, expectations and even variances
- Model order selection/comparison can be implemented easily (available in a future release of the package)

[1] at the time of writing this document, only first order models are supported

The hyperparameters determine the form of the prior distribution, which is a product of Dirichlet distributions

$$P(\theta) = \prod_{u \in \mathcal{A}} \Big\{ \frac{\Gamma(\alpha(u))}{\prod_{s \in \mathcal{A}} \Gamma(\alpha(u,s))} \prod_{s \in \mathcal{A}} p(s|u)^{\alpha(u,s))-1} \Big\}$$

where $\Gamma(.)$ is the Gamma function. The hyperparameters are specified using the `hyperparam` argument in the `markovchainFit` function. If this argument is not specified, then a default value of 1 is assigned to each hyperparameter resulting in the prior distribution of each chain parameter to be uniform over $[0, 1]$.

Given the likelihood and the prior as described above, the **evidence** $P(D)$ is simply given by

$$P(D) = \int P(D|\theta) P(\theta) d\theta$$

which simplifies to

$$P(D) = \prod_{u \in \mathcal{A}} \Big\{ \frac{\Gamma(\alpha(u))}{\prod_{s \in \mathcal{A}} \Gamma(\alpha(u,s))} \frac{\prod_{s \in \mathcal{A}} \Gamma(n(u,s) + \alpha(u,s))}{\Gamma(\alpha(u) + n(u))} \Big\}$$

Using Bayes' theorem, the **posterior** now becomes (thanks to the choice of conjugate priors)

$$P(\theta|D) = \prod_{u \in \mathcal{A}} \Big\{ \frac{\Gamma(n(u) + \alpha(u))}{\prod_{s \in \mathcal{A}} \Gamma(n(u,s) + \alpha(u,s))} \prod_{s \in \mathcal{A}} p(s|u)^{n(u,s)+\alpha(u,s))-1} \Big\}$$

Since this is again a product of Dirichlet distributions, the marginalised distribution of a particular parameter $P(s|u)$ of our chain is given by

$$P(s|u) \sim Beta(n(u,s) + \alpha(u,s), n(u) + \alpha(u) - n(u,s) - \alpha(u,s))$$

Thus, the MAP estimate $\hat{\theta}$ is given by

$$\hat{\theta} = \Big\{ \frac{n(u,s) + \alpha(u,s) - 1}{n(u) + \alpha(u) - |\mathcal{A}|}, s \in \mathcal{A}, u \in \mathcal{A} \Big\}$$

The function also returns the expected value, given by

$$\mathrm{E}_{\mathrm{post}} p(s|u) = \Big\{ \frac{n(u,s) + \alpha(u,s)}{n(u) + \alpha(u)}, s \in \mathcal{A}, u \in \mathcal{A} \Big\}$$

The variance is given by

$$\mathrm{Var}_{\mathrm{post}} p(s|u) = \frac{n(u,s) + \alpha(u,s)}{(n(u) + \alpha(u))^2} \frac{n(u) + \alpha(u) - n(u,s) - \alpha(u,s)}{n(u) + \alpha(u) + 1}$$

The square root of this quantity is the standard error, which is returned by the function.

The confidence intervals are constructed by computing the inverse of the beta integral.

Predictive distribution

Given the old data set, the probability of observing new data is $P(D'|D)$ where D' is the new data set. Let $m(u,s), m(u)$ denote the corresponding counts for the new data. Then,

$$P(D'|D) = \int P(D'|\theta) P(\theta|D) d\theta$$

We already know the expressions for both quantities in the integral and it turns out to be similar to evaluating the evidence

$$P(D'|D) = \prod_{u \in \mathcal{A}} \Big\{ \frac{\Gamma(\alpha(u))}{\prod_{s \in \mathcal{A}} \Gamma(\alpha(u,s))} \frac{\prod_{s \in \mathcal{A}} \Gamma(n(u,s) + m(u,s) + \alpha(u,s))}{\Gamma(\alpha(u) + n(u) + m(u))} \Big\}$$

Choosing the hyperparameters

The hyperparameters model the shape of the parameters' prior distribution. These must be provided by the user. The package offers functionality to translate a given prior belief transition matrix into the hyperparameter matrix. It is assumed that this belief matrix corresponds to the mean value of the parameters. Since the relation

$$\mathrm{E}_{\mathrm{prior}} p(s|u) = \frac{\alpha(u,s)}{\alpha(u)}$$

holds, the function accepts as input the belief matrix as well as a scaling vector (serves as a proxy for $\alpha(.)$) and proceeds to compute $\alpha(.,.)$.

Alternatively, the function accepts a data sample and infers the hyperparameters from it. Since the mode of a parameter (with respect to the prior distribution) is proportional to one less than the corresponding hyperparameter, we set

$$\alpha(u,s) - 1 = m(u,s)$$

where $m(u,s)$ is the $u- > s$ transition count in the data sample. This is regarded as a 'fake count' which helps $\alpha(u,s)$ to reflect knowledge of the data sample.

Usage and examples

```
library(markovchain)
weatherStates <- c("sunny", "cloudy", "rain")
byRow <- TRUE
weatherMatrix <- matrix(data = c(0.7, 0.2, 0.1,
                                 0.3, 0.4, 0.3,
                                 0.2, 0.4, 0.4),
                        byrow = byRow, nrow = 3,
                        dimnames = list(weatherStates, weatherStates))
mcWeather <- new("markovchain", states = weatherStates,
                 byrow = byRow, transitionMatrix = weatherMatrix,
                 name = "Weather")
weathersOfDays <- rmarkovchain(n = 365, object = mcWeather, t0 = "sunny")
```

For the purpose of this section, we shall continue to use the weather of days example introduced in the main vignette of the package (reproduced above for convenience).

Let us invoke the fit function to estimate the MAP parameters with 92% confidence bounds and hyperparameters as shown below, based on the first 200 days of the weather data. Additionally, let us find out what the probability is of observing the weather data for the next 165 days. The usage would be as follows

```
hyperMatrix<-matrix(c(1, 1, 2,
                      3, 2, 1,
                      2, 2, 3),
                    nrow = 3, byrow = TRUE,
                    dimnames = list(weatherStates,weatherStates))
markovchainFit(weathersOfDays[1:200], method = "map",
               confidencelevel = 0.92, hyperparam = hyperMatrix)
```

```
## Warning in markovchainFit(weathersOfDays[1:200], method = "map", confidencelevel = 0.92, :
## 'estimate' is the MAP set of parameters where as 'expectedValue'
## is the expectation of the parameters with respect to the posterior.
## The confidence intervals are given for 'estimate'.

## $estimate
## Bayesian Fit
##  A  3 - dimensional discrete Markov Chain with following states
##  cloudy rain sunny
##  The transition matrix   (by rows)  is defined as follows
##           cloudy      rain     sunny
## cloudy 0.3970588 0.2647059 0.3382353
## rain   0.3333333 0.5303030 0.1363636
## sunny  0.2465753 0.1780822 0.5753425
##
##
## $expectedValue
##           cloudy      rain     sunny
## cloudy 0.3943662 0.2676056 0.3380282
## rain   0.3333333 0.5217391 0.1449275
## sunny  0.2500000 0.1842105 0.5657895
##
## $standardError
##            [,1]       [,2]       [,3]
## [1,] 0.05759551 0.05217397 0.05574808
## [2,] 0.05634362 0.05970492 0.04207537
## [3,] 0.04934638 0.04417748 0.05648488
##
## $confidenceInterval
## $confidenceInterval$confidenceLevel
## [1] 0.92
##
## $confidenceInterval$lowerEndpointMatrix
##           [,1]      [,2]      [,3]
## [1,] 0.3083874 0.1781098 0.2500184
## [2,] 0.2440591 0.4373388 0.0000000
## [3,] 0.1631835 0.0941740 0.4856027
##
## $confidenceInterval$upperEndpointMatrix
##           [,1]      [,2]      [,3]
## [1,] 0.5226492 0.3598184 0.4485619
## [2,] 0.4443330 1.0000000 0.2071556
## [3,] 0.3349244 0.2524594 1.0000000
##
##
## $logLikelihood
## [1] -199.7715
```

```
predictiveDistribution(weathersOfDays[1:200],
                       weathersOfDays[201:365],hyperparam = hyperMatrix)
```

```
## [1] -158.2507
```

The results should not change after permuting the dimensions of the matrix.

```r
hyperMatrix2<- hyperMatrix[c(2,3,1), c(2,3,1)]
markovchainFit(weathersOfDays[1:200], method = "map",
               confidencelevel = 0.92, hyperparam = hyperMatrix2)
```

```
## Warning in markovchainFit(weathersOfDays[1:200], method = "map", confidencelevel = 0.92, :
## 'estimate' is the MAP set of parameters where as 'expectedValue'
## is the expectation of the parameters with respect to the posterior.
## The confidence intervals are given for 'estimate'.
```

```
## $estimate
## Bayesian Fit
##  A  3 - dimensional discrete Markov Chain with following states
##  cloudy rain sunny
##  The transition matrix   (by rows)  is defined as follows
##           cloudy      rain     sunny
## cloudy 0.3970588 0.2647059 0.3382353
## rain   0.3333333 0.5303030 0.1363636
## sunny  0.2465753 0.1780822 0.5753425
##
##
## $expectedValue
##           cloudy      rain     sunny
## cloudy 0.3943662 0.2676056 0.3380282
## rain   0.3333333 0.5217391 0.1449275
## sunny  0.2500000 0.1842105 0.5657895
##
## $standardError
##            [,1]       [,2]       [,3]
## [1,] 0.05759551 0.05217397 0.05574808
## [2,] 0.05634362 0.05970492 0.04207537
## [3,] 0.04934638 0.04417748 0.05648488
##
## $confidenceInterval
## $confidenceInterval$confidenceLevel
## [1] 0.92
##
## $confidenceInterval$lowerEndpointMatrix
##           [,1]      [,2]      [,3]
## [1,] 0.3083874 0.1781098 0.2500184
## [2,] 0.2440591 0.4373388 0.0000000
## [3,] 0.1631835 0.0941740 0.4856027
##
## $confidenceInterval$upperEndpointMatrix
##           [,1]      [,2]      [,3]
## [1,] 0.5226492 0.3598184 0.4485619
## [2,] 0.4443330 1.0000000 0.2071556
## [3,] 0.3349244 0.2524594 1.0000000
##
##
## $logLikelihood
## [1] -199.7715
```

```
predictiveDistribution(weathersOfDays[1:200],
                       weathersOfDays[201:365],hyperparam = hyperMatrix2)
```

```
## [1] -158.2507
```

Note that the predictive probability is very small. However, this can be useful when comparing model orders.

Suppose we have an idea of the (prior) transition matrix corresponding to the expected value of the parameters, and have a data set from which we want to deduce the MAP estimates. We can infer the hyperparameters from this known transition matrix itself, and use this to obtain our MAP estimates.

```
inferHyperparam(transMatr = weatherMatrix, scale = c(10, 10, 10))
```

```
## $scaledInference
##        cloudy rain sunny
## cloudy      4    3     3
## rain        4    4     2
## sunny       2    1     7
```

Alternatively, we can use a data sample to infer the hyperparameters.

```
inferHyperparam(data = weathersOfDays[1:15])
```

```
## $dataInference
##        cloudy rain sunny
## cloudy      4    2     2
## rain        2    4     3
## sunny       2    3     1
```

In order to use the inferred hyperparameter matrices, we do

```
hyperMatrix3 <- inferHyperparam(transMatr = weatherMatrix, scale = c(10, 10, 10))
hyperMatrix3 <- hyperMatrix3$scaledInference

hyperMatrix4 <- inferHyperparam(data = weathersOfDays[1:15])
hyperMatrix4 <- hyperMatrix4$dataInference
```

Now we can safely use *hyperMatrix3* and *hyperMatrix4* with *markovchainFit* (in the *hyperparam* argument).

Supposing we don't provide any hyperparameters, then the prior is uniform. This is the same as maximum likelihood.

```
data(preproglucacon, package = "markovchain")
preproglucacon <- preproglucacon[[2]]
MLEest <- markovchainFit(preproglucacon, method = "mle")
MAPest <- markovchainFit(preproglucacon, method = "map")
```

```
## Warning in markovchainFit(preproglucacon, method = "map"):
## 'estimate' is the MAP set of parameters where as 'expectedValue'
## is the expectation of the parameters with respect to the posterior.
## The confidence intervals are given for 'estimate'.
```

```
MLEest$estimate
```

```
## MLE Fit
##  A  4 - dimensional discrete Markov Chain with following states
##  A C G T
##  The transition matrix   (by rows)  is defined as follows
##           A         C          G         T
## A 0.3585271 0.1434109 0.16666667 0.3313953
## C 0.3840304 0.1558935 0.02281369 0.4372624
## G 0.3053097 0.1991150 0.15044248 0.3451327
## T 0.2844523 0.1819788 0.17667845 0.3568905
```

```
MAPest$estimate
```

```
## Bayesian Fit
##  A  4 - dimensional discrete Markov Chain with following states
##  A C G T
##  The transition matrix   (by rows)  is defined as follows
##           A         C          G         T
## A 0.3585271 0.1434109 0.16666667 0.3313953
## C 0.3840304 0.1558935 0.02281369 0.4372624
## G 0.3053097 0.1991150 0.15044248 0.3451327
## T 0.2844523 0.1819788 0.17667845 0.3568905
```