FD3447/DD2447 - Statistical Methods in Applied Computer Science, Fall 2020

Assignment 2

Adriano Mundo, Daniele Massaro

January 15, 2021

Contents

0 Introduction

The report describes the code which has been implemented and the obtained results in order to assess to second assignment of the course Statistical Methods in Applied Computer Science. For each problem the assumptions, the details of implementation and the final results are shown.

Python3 and R3.6 programming languages have been used for the implementation. Attached with this report it is possible to find the code. Each subsection is labeled with the corresponding question [Qi].

1 Gibbs sampler for the magic word

1.1 Problem description

Let consider N sequences of length M over an alphabet K. In this case the alphabet is made of the 1-digit prime number, so $K = 4$ with the elements $2, 3, 5, 7$. Each sequence has a *magic* word of length W hidden in the *background*.

- The starting position of the magic world is sampled uniformly from $[M - W + 1]$.
- The $j : th$ positions in the magic words are sampled from $q_j(x) = Cat(x|\theta_j)$, where θ_j has a $Dir(\theta_j|\alpha)$ prior.
- All other positions are sampled from $q(x) = Cat(x|\theta)$, where θ has a $Dir(\theta_j|\alpha')$ prior.

The posterior $p(r_1, r_2, ..., r_N|D)$ has to evaluated, where D is a set of sequences and r_n is the start position of the magic word. A Gibbs sampler has been implemented for estimating the posterior after having observed N sequences. Since the samples is collapsed we do not know, in principle, the hyperparameters α and α'.

1.2 Gibbs Sampler [Q1]

The Gibbs sampler code generates a list of sequences and a list of starting positions. The following parameters need to be specified:

- W: length of the word.
- N: number of sequences.
- M: length of the sequences.
- α: parameter of the magic-word Dirichlet prior distribution.
- α': parameter of the background Dirichlet prior distribution.
- K: length of the alphabet and the alphabet itself $(2, 3, 5, 7)$.

The code defines the magic word and background distribution and then builds up a sequence where the initial position is defined. The parameters θ and θ' are sampled randomly from a prior Dirichlet distribution with α and α' parameters respectively.

For example considering $W = 3$, $N = 5$, $M = 10$, $\alpha = [1, 1, 1, 1]$, $\alpha' = [1, 7, 12, 2]$ and the alphabet$(2, 3, 5, 7)$, we get the sequences:

$$\begin{array}{c} 7,5,3,2,2,2,\mathbf{2},2,5,7 \\ \mathbf{5},3,5,2,7,7,3,2,2,7 \\ 2,2,5,\mathbf{2},3,5,5,2,7,2 \\ 7,2,2,2,\mathbf{2},3,5,5,5,5 \\ 2,2,7,2,\mathbf{2},3,5,2,2,2 \end{array} \tag{1}$$

And the starting positions:

$$7,0,4,5,5 \tag{2}$$

1.3 Synthetic data simulation and posterior estimation [Q1]

Firstly the synthetic set of data is generated using the previous introduced Gibbs sampler. Hence the complete sequences and the correct initial positions are defined. The parameters settings are:

- $W = 10$
- $N = 5$
- $M = 30$
- $\alpha = [0.1, 0.1, 0.1, 0.1]$
- $\alpha' = [9, 7, 20, 2]$
- $K = 4$ and the alphabet $(2, 3, 5, 7)$

The numerical simulations are parameters are set in order to guarantee convergence and avoid the *burn-in* period. Several tests have been performed, which are presented in the next session, and eventually the chosen parameters are:

- Number of iterations: 100
- Burn-in period: 50
- Lag length: 10

The sampler is run for 100 iterations and first half is discard to avoid the transient burn-in period, which is a conservative choice for the MCMC samples. This is relevant since the randomly set initial values could be far from the high density regions of the stationary distribution, so the early stages of the simulation gives sampled values that are unlikely to occur. Several tests have been performed to evaluate how the lag affects the convergence of the results (see below) and at the end 10 has been adopted.

Let consider these settings and evaluate the accuracy of our estimation. The result can be sensible to our α and α' parameters, but usually shows a quite good agreement. The Gibbs sampler is used to estimate the correct initial positions $(19, 0, 2, 10, 14)$ and counting the most probable correct initial positions, after the burn-in period with a lag equal 10:

$$19,0,2,10,14 \tag{3}$$

The accuracy is equal to 80% looking at the final estimation of the correct initial position, a little bit less considering the "historical" accuracy, hence counting the number of times that the initial positions are estimated corrected even considering the initial transient (around 69%). Even if the last sequence is not estimated correctly, the values is still very close (14 instead of 15).

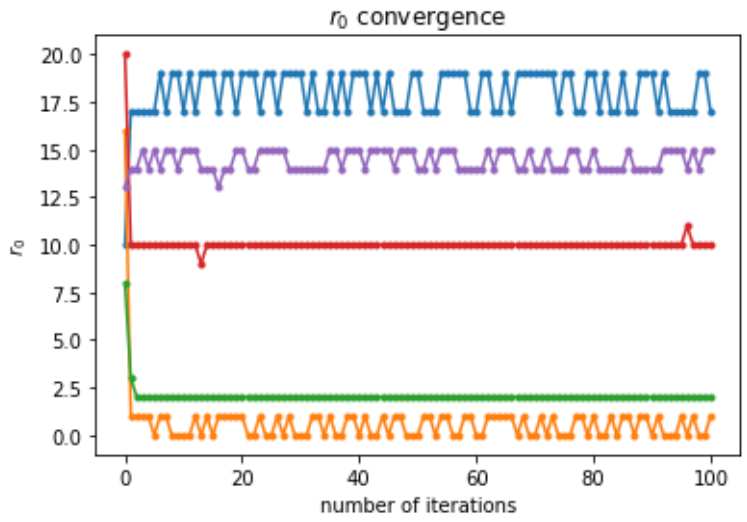


Figure 1: Estimated initial positions as function of number of iterations.

As Fig.9 shows the initial burn-in period is very limited and avoiding the first half of total iterations number is very conservative. By the way in some sequences an oscillating behaviour around the correct initial position is observed. Increasing the number of iterations does not seem to affect it. In order to estimate the convergence level of our series the mean value ($\bar{r}_0$) and standard deviation (σ) have been estimated. For each sequences they have very small values, especially the variance, guaranteeing a small oscillation around the correct value.

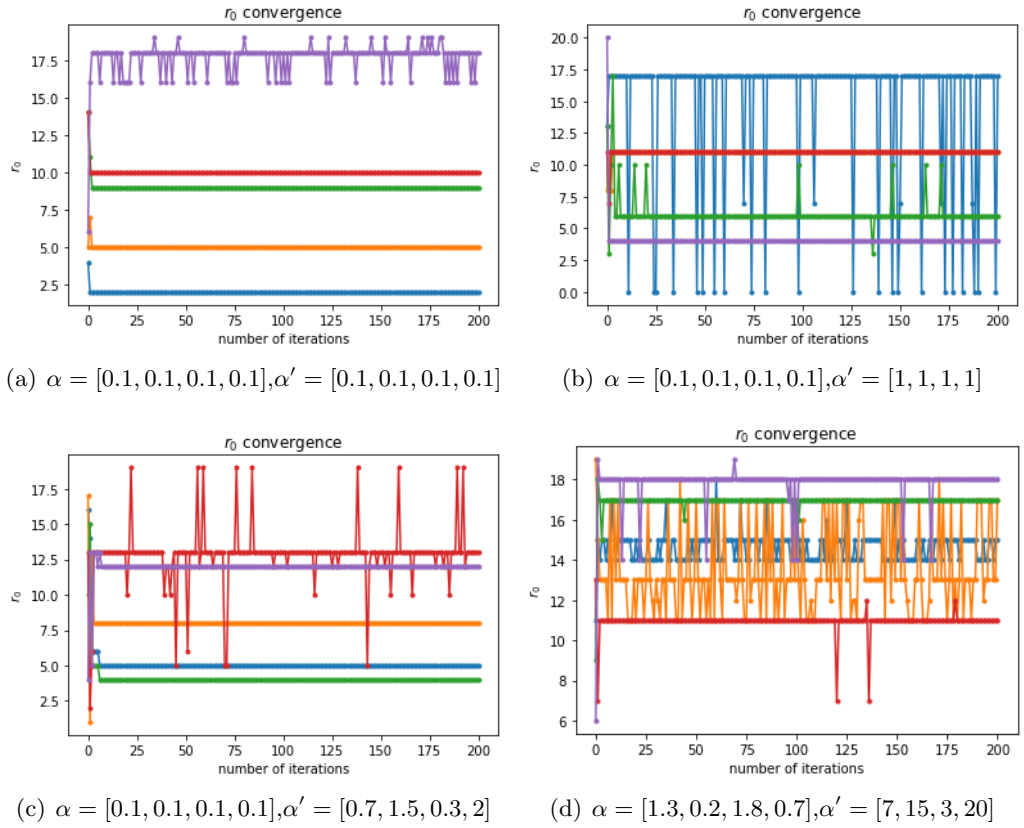
$$\begin{aligned} \bar{r}_0 &= 18.10, \quad \sigma = 0.99 \\ \bar{r}_0 &= 0.51, \quad \sigma = 0.49 \\ \bar{r}_0 &= 2.01, \quad \sigma = 0.09 \\ \bar{r}_0 &= 10.01, \quad \sigma = 0.14 \\ \bar{r}_0 &= 14.46, \quad \sigma = 0.51 \end{aligned} \tag{4}$$

1.4 Convergence analysis [Q1]

Several parameters can potentially affect the validity and the convergence of the results. Tests have been carried out in order to evaluate their influence.

Firstly let consider the parameters α and α' of the prior Dirichlet distribution for θ variable of the magic and background positions. All other parameters are fixed ($W = 10$, $N = 5$, $M = 30$, $W = 4$, $it = 200$, $bn = 100$, $lg = 10$ and the final accuracy 80%). Please observe that it is the number of iterations, bn is the number of iterations for the burn-in period and lg is the sampling lag. Four test cases with four different α-α' have been carried out. Fig.2 shows as when α and α' have values different and not symmetric (for each letter of the alphabet) higher convergence is guaranteed, due to different assigned prior probability, the Gibbs sampler is able to estimate in a more accurate way the correct position. The case C with even larger α' values guarantees smaller oscillations and higher convergence, hence this last settings have been adopted.

The accuracy is now taken into account. For a generic α-α' couple different accuracy can be achieved. Setting all other parameters the "final" and "historical" accuracy can show different behaviours: especially when very large oscillations are present at the very beginning, typically they are not damped out anymore, affecting the overall result. Fig.3 shows several possible configurations



(a) $\alpha = [0.1, 0.1, 0.1, 0.1]$,$\alpha' = [0.1, 0.1, 0.1, 0.1]$

(b) $\alpha = [0.1, 0.1, 0.1, 0.1]$,$\alpha' = [1, 1, 1, 1]$

(c) $\alpha = [0.1, 0.1, 0.1, 0.1]$,$\alpha' = [0.7, 1.5, 0.3, 2]$

(d) $\alpha = [1.3, 0.2, 1.8, 0.7]$,$\alpha' = [7, 15, 3, 20]$

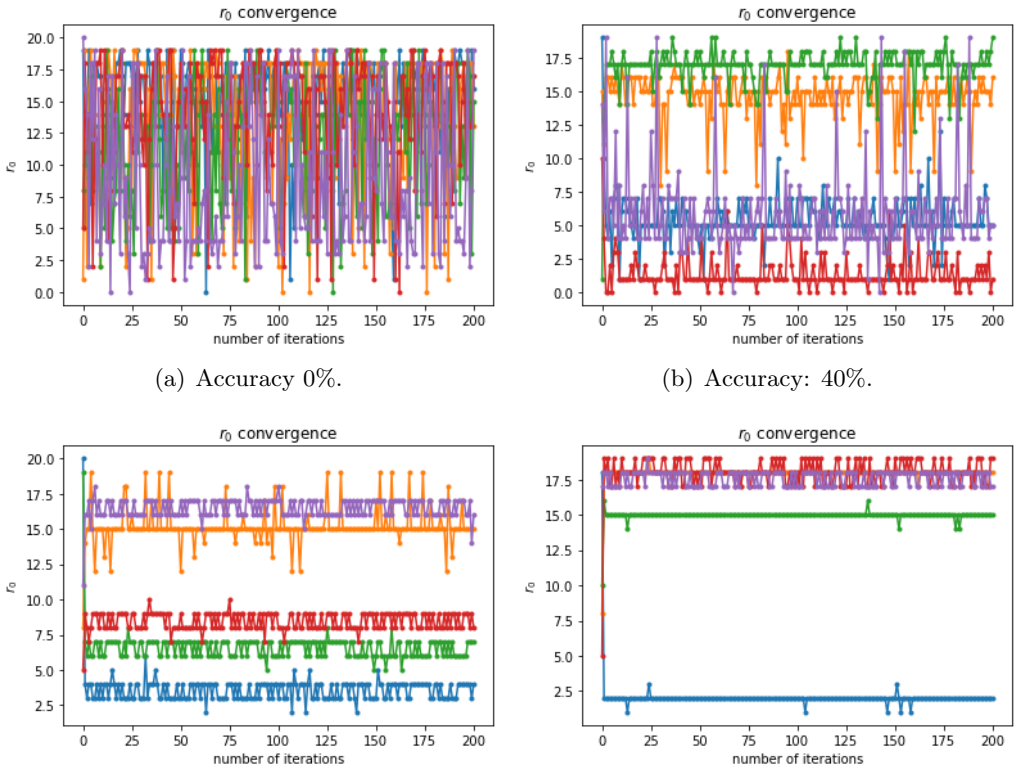
Figure 2: Initial positions estimation for different α and α' couples. For each couple equal/different and symmetric/not symmetric distributions have been considered. Other settings: $W = 10$, $N = 5$, $M = 30$, $W = 4$, $it = 200$, $bn = 100$, $lg = 10$ and the final accuracy 80%.

which can occur when parameters are not properly set.

Eventually numerical parameters are taken into account. In all previous cases the burn-in initial period does not show a great relevance, since in few iterations the stationary state is reached. As expected Fig.6-A shows that considering $it = 200$ and $lg = 10$ and increasing the burn-in period from $bn = 10$ to $bn = 100$ (half of the simulation length), does not affect significantly the simulations accuracy, even if we slightly improve it. At the same way the sampling lag does not guarantee a significant increase in the accuracy. Obviously the smaller lag gives higher accuracy, but the gain is not so relevant (from 0.8975% to 0.8800%). Moreover an other aspect should be taken into account: smaller lag gives higher computational cost. Eventually let consider the total number of iterations. Fig. 6-C shows an (almost) monotonically increase of the accuracy, but only of few percentage points. On the other hand it significantly affects the computational cost, hence a reasonable number of iterations can be chosen between 200 and 400.

1.5 Database testing [Q2]

In the end the given database is tested estimating the most probable initial positions. The main parameters are given: $N = 50$, $M = 40$, $W = 6$ and $K = 7$. The alphabet is made by the first 7 integer real number $(0, 1, 2, 3, 4, 5, 6)$. The Dirichlet parameters are defined as $\alpha = [0.9, 0.9, 0.9, 0.9, 0.9, 0.9, 0.9]$ and $\alpha' = [1, 1, 1, 1, 1, 1, 1]$. Since these parameters are given,



(a) Accuracy 0%.

(b) Accuracy: 40%.

(c) Accuracy 60%.

(d) Accuracy 100%.

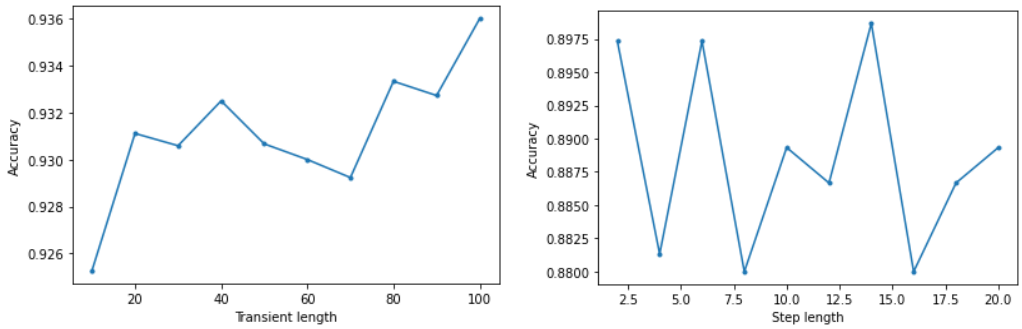
Figure 3: The figure shows how the parameters choice can affect the final accuracy and in case compromise the code validity. For several set of parameters different accuracy is reached (from 0% to 100%). The variation of the initial estimated starting position is showed. When the sequences reach convergence the initial burn-in lasts only few iterations.

only numerical ones can be set according to previous convergence results. Once set the random seed, the estimated initial starting position for the first 10 sequences results:

$$12, 24, 27, 12, 33, 14, 10, 29, 4, 12 \tag{5}$$

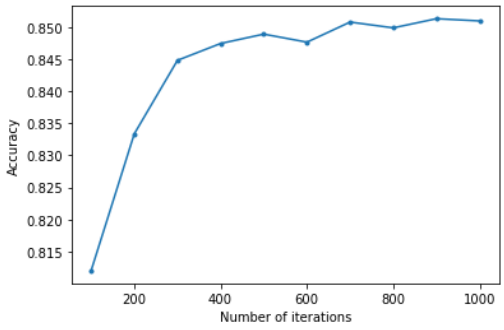
The numerical parameters are set equal to $it = 400$, $bn = 200$, $lg = 10$. The lag equal 10 guarantees a good balance between performances and accuracy. The burn-in period is expired after some iterations, but in order to be conservative only the second half of the entire data set is taken. Eventually the number of iterations is set to 400, since up to 1000 the result is not improved significantly (see 6).

The test has been repeated several times in order to guarantee reliability and with several parameters, as shown in the previous convergence analysis. The initial position list has always been confirmed in each test. The original database was even larger, containing 50 sequences and additional tests including more sequences have been performed.



(a) Accuracy as function of the initial burn-in period.

(b) Accuracy as function of the sampling lag.



(c) Accuracy as function of the number of iterations.

Figure 4: The figures show how the total number of iterations, the sampling lag and the initial burn-in affect the overall accuracy. Other parameters are fixed.

2 SMC for the stochastic volatility model

2.1 Problem description

Given the following stochastic volatility model with $t = 1, ..., T$ and vector parameters $\theta = \{\phi, \sigma, \beta\}$:

$$X_1 \sim \mathcal{N}(x_1|0, \sigma^2) \tag{6}$$

$$X_t|(X_{t-1} = x_{t-1}) \sim \mathcal{N}(x_t|\phi x_{t-1}, \sigma^2) \tag{7}$$

$$Y_t|(X_t = x_t) \sim \mathcal{N}(y_t|0, \beta^2 exp(x_t)) \tag{8}$$

X_t is the latent variable which underlyins volatility and Y_t is the observed scaled log-returns.

The objectives of this task are to simulate data generation with fixed parameters, implement the sequential importance sampling and the bootstrap particle filter with two resampling techniques: multinomial and stratified resampling. Finally, a comparison between the three techniques is made.

2.2 Data Generation [Q3]

Attached to the report there exist a file SV_data_generation.R which contains a generate_data function which generate the stochastic volatility data, hence $x_{1:t}$ and $y_{1:t}$ with $T = 100$ and fixed $\phi = 0.91, \sigma = 0.16, \beta = 0.64$. The function is parametric and can be used with different parameters. Fig. 5 shows the result from the simulation process.

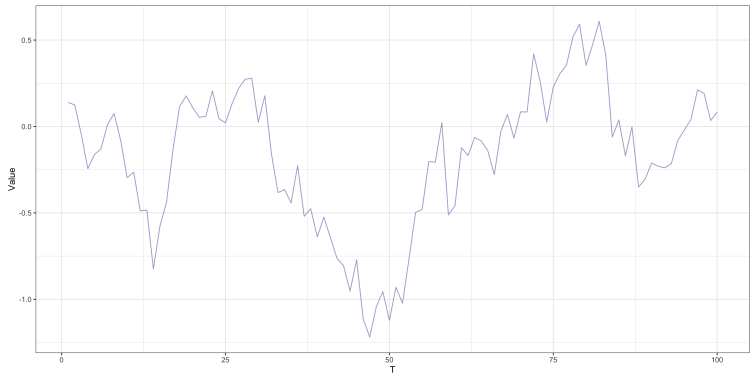


Figure 5: stochastic volatility data generated

2.3 Sequential Importance Sampling [Q4]

(i) The choice of the proposal is $q(x_t|y_{1:t}) = \sum_i w_{t-1}^i p(x_t|x_{t-1}^i)$ because it allows to compute weights by using known distributions that defines $w_t = p(y_t|x_t)$. The target $p(x_{1:t}|y_{1:t})$ has N particles $\{x_{1:t}^i\}_{i=1}^N$, which are obtained using the proposal distribution $q(x_t|y_{1:t})$. Each particle has its importance weight:

$$w_t^i = \frac{p(x_{1:t}^i|y_{1:t})}{q(x_{1:t}^i|y_{1:t})} \tag{9}$$

The target distribution is known up to a normalizing constant. The normalized weights can be obtained from the normal (non-normalized) weights in this way: $w_t^i = \frac{\tilde{w}_t^i}{\sum_i \tilde{w}_t^i}$. It is possible to use sequential relations, hence the proposal distribution can be factorised in this way:

$$q_t(x_{1:t}|y_{1:t}) = q_t(x_t|x_{t-1}, y_t)q_{t-1}(x_{1:t-1}|y_{1:t-1}) = q_1(x_1|y_1)\prod_k q_k(x_k|x_{k-1}, y_k) \tag{10}$$

The weights can be factorised too:

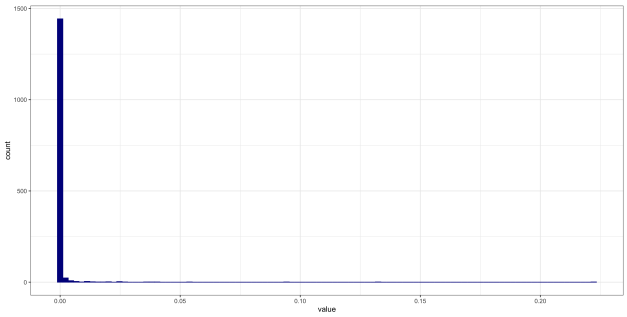
$$w_t^i = \frac{p(x_{1:t}^i|y_{1:t})}{q(x_{1:t}^i|y_{1:t})} \propto \frac{p(y_t|x_t^i)p(x_t^i|x_{t-1}^i)}{q_t(x_t^i|x_{t-1}^i, y_t)} \frac{p(x_{1:t-1}^i|y_{1:t-1})}{q_{t-1}(x_{1:t-1}^i|y_{1:t-1})} = \frac{p(y_t|x_t^i)p(x_t^i|x_{t-1}^i)}{q_t(x_t^i|x_{t-1}^i, y_t)} w_{t-1}^i \tag{11}$$

(ii) In the *Sequential Importance Sampling (SIS)*, where $x_{100} = 0.08305107$ in the test data, the point estimate is $\hat{x}_{100} = 0.2671678$ and the MSE of the estimate to the truth value when N is increased is $MSE = 0.03389897$. Fig. 6b shows the MSE with respect to the value of N

(iii) In the *Sequential Importance Sampling (SIS)*, the empirical Var of the normalized weights is equal to $Var(\hat{w}_{100}) = 4.890004 \times 10^{-5}$. Fig 6a shows the histogram of weights, which suffer from weights degeneracy.

2.4 Multinomial Bootstrap Particle Filter [Q5]

(ii) In the *Bootstrap Particle Filter (BPF)* with *Multinomial Re-Sampling*, where $x_{100} = 0.08305107$ in the test data, the point estimate is $\hat{x}_{100} = 0.09018866$ and the MSE of the estimate to the truth value when N is increased is $MSE = 0.05094529$. Fig. 7b shows the MSE with respect to the value of N.

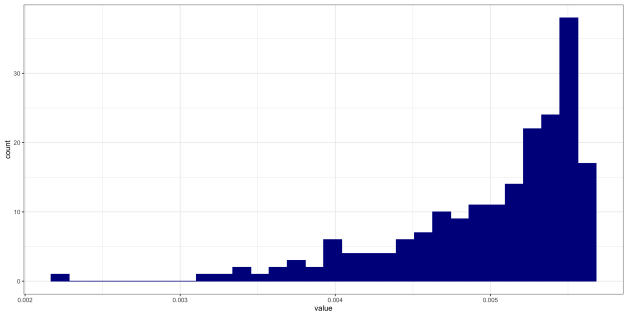


(a) SIS normalized weights histogram

(b) SIS mean square error with different N values

Figure 6: The figures shows the SIS mean square error and weights.

(iii) In the *Bootstrap Particle Filter (BPF)* with *Multinomial Re-Sampling*, the empirical Var of the normalized weights is equal to $Var(\hat{w}_{100}) = 6.290525 \times 10^{-9}$. Fig 7a shows the histogram of weights, which does not suffer anymore from the weights degeneracy problem.



(a) BPF multinomial normalized weights histogram

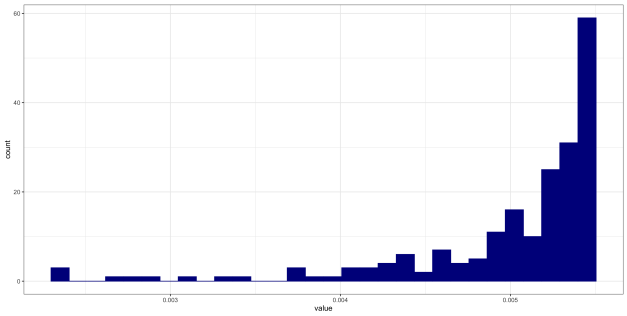
(b) BPF multinomial mean square error with different N values

Figure 7: The figures shows the BPF multinomial mean square error and weights.

2.5 Stratified Bootstrap Particle Filter [Q6]

(ii) In the *Bootstrap Particle Filter (BPF)* with *Stratified Re-Sampling*, where $x_{100} = 0.08305107$ in the test data, the point estimate is $\hat{x}_{100} = 0.2460023$ and the MSE of the estimate to the truth value when N is increased is $MSE = 0.02635846$. Fig. 8b shows the MSE with respect to the value of N.

(iii) In the *Bootstrap Particle Filter (BPF)* with *Stratified Re-Sampling*, the empirical Var of the normalized weights is equal to $Var(\hat{w}_{100}) = 4.358215 \times 10^{-9}$. Fig 8a shows the histogram of weights, which does not suffer anymore from the weights degeneracy problem.



(a) BPF stratified normalized weights histogram (b) BPF stratified mean square error with different N values

Figure 8: The figures shows the BPF stratified mean square error and weights.

Technique	$\hat{x}_{100}$	MSE	$Var(\hat{w}_{100})$
SIS	0.2671678	0.03389897	4.890004×10^{-5}
BPF Multinomial	0.09018866	0.05094529	6.290525×10^{-9}
BPF Stratified	0.2460028	0.02635846	4.358215×10^{-9}

Table 1: Table for techniques comparison

2.6 Comparison [Q7]

SIS suffers from weight degeneracy, while BPF fixes the problem. In general, SIS performs (a little) worse than BPF because they solve the aforementioned problem with resampling. Stratified sampling seems to be the best in according with the MSE but the performance are very similar to the multinomial bootstrap filter. Fig.9 shows a visual comparison between the three techniques.

3 Stochastic volatility unknown parameters part I

3.1 Problem Description

The scenario of this task is very close the previous about SMC for the stochastic volatility model, but there are some unknown parameters, which are σ and β, while $\phi = 0.91$. We need to simulate the data and create a grid with a number of combinations of both σ and β and by leveraging

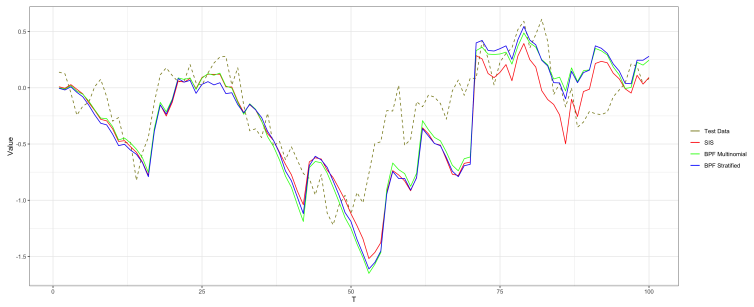


Figure 9: SIS and BPF comparison

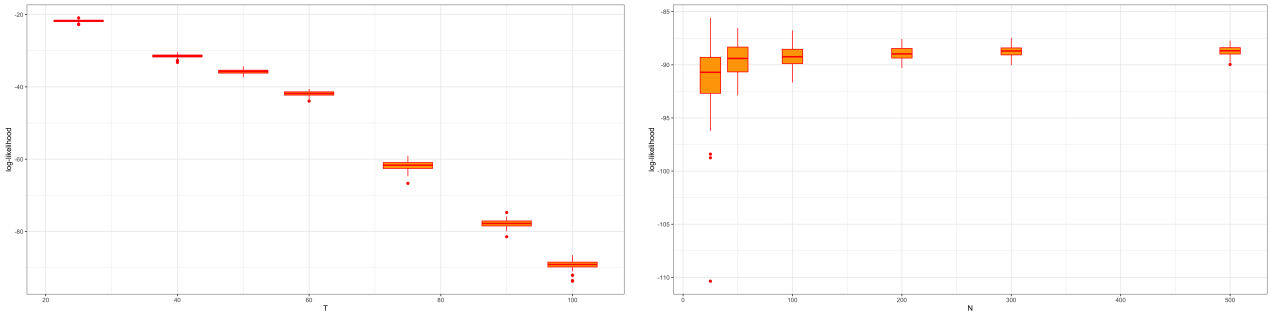
on the BPF with Multinomial re-sampling estimate the marginal likelihood $p_\theta(y_{1:T})$ for each pair. Once that, we need to find the best combination. Another interesting point is to understand the relationship between T, N and the Var of the log-likelihood estimation since BPF can be used as log-likelihood estimator. Finally, an implementation and evidence of Particle Metropolis-Hastings (PMH) is provided.

3.2 Parameter Estimation [Q8]

In our experiment we evaluated 625 pairs of (σ, β) between 0 and 2 with difference of 0.08. SMC was run 10 times for each combination, the mean value of the log-likelihood was compared among the pairs. The best parameters combination we obtained is $\sigma = 0.161, \beta = 0.641$ with a $log\hat{p}(y_{1:T}) = -87.7$. The combination is exactly equal to the parameters value we used to generated the data $\sigma = 0.16, \beta = 0.64$.

3.3 Log-likelihood Variance [Q9]

The study suggests that the variance of the log-likelihood increases with an increase in T. The main reason is the temporal horizon which increase the difficulty of a right prediction as we try to predict something far in the future. Instead, the variance of the log-likelihood decreases with an increase in the value of N. The reason is that an increase in the number of particles equals to an increase in the number of samples which can give a more confident estimation. Fig. 10 shows two box plot of the results described in this paragraph.



(a) T variation, N = 80

(b) N variation, T = 100

Figure 10: Variance of the log-likelihood with T and N variation.

3.4 Particle Metropolis Hastings [Q10]

Our implementation tested the PMH sampler with 15000 iterations and discarded the first 5000 for the burn-in effect.

(i) The proposal distribution for β, σ are:

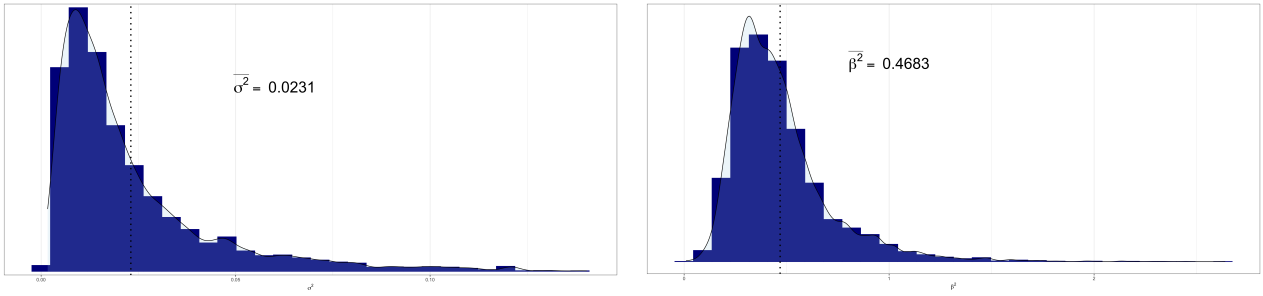
- $\sigma^2_{current} = max\{0.0001, \mathcal{N}(\sigma^2_{previous}, 0.01)\}$
- $\beta^2_{current} = max\{0.01, \mathcal{N}(\beta^2_{previous}, 0.2)\}$

The two parameters (σ, β) has to be positive, hence we opted for the max function in order to not waste cycles on obtaining positive values which are assigned as small values over zero.

Moreover, it does not have to be equal to zero for avoiding the situation where all particles are mutually identical due to zero Var or Std.

The Metropolis-Hastings acceptance ratio obtained is 0.4444.

(ii) To prove the evidence of success, we show the posterior distribution for σ^2 and β^2. Fig. 11 shows the plot of the two distributions.



(a) σ^2 posterior distribution

(b) β^2 posterior distribution

Figure 11: Posterior distribution of PMH sampler

The posterior mean gives value close to the original value of $\sigma = 0.16$. Indeed, $\hat{\sigma} = \sqrt{0.0231} = 0.1519$. On the same line, the posterior mean gives $\hat{\beta} = \sqrt{0.4683} = 0.6843$. The real value is $\beta = 0.64$. The result obtained for σ are not good as for β but still remain a valid result. Moreover, the result are realistic since the experiment was conducted with $N = 10000$ samples after discarding 5000 samples for the burn-in.

4 Stochastic volatility unknown parameters part II

4.1 Problem Description

In this task, the setup is very close the previous two in 2.2 and 2.3 section. In this case, we have some unknown parameters and the objective is to implement a particle Gibbs sampler to compute the posterior distribution $p(\sigma^2, \beta^2|\phi, y_{1:T})$ by leveraging on the conditional SMC. We obtained the marginal distribution and verified the results with respect to the Particle Metropolis-Hastings (PMH) sampler.

4.2 Conditional SMC [Q11]

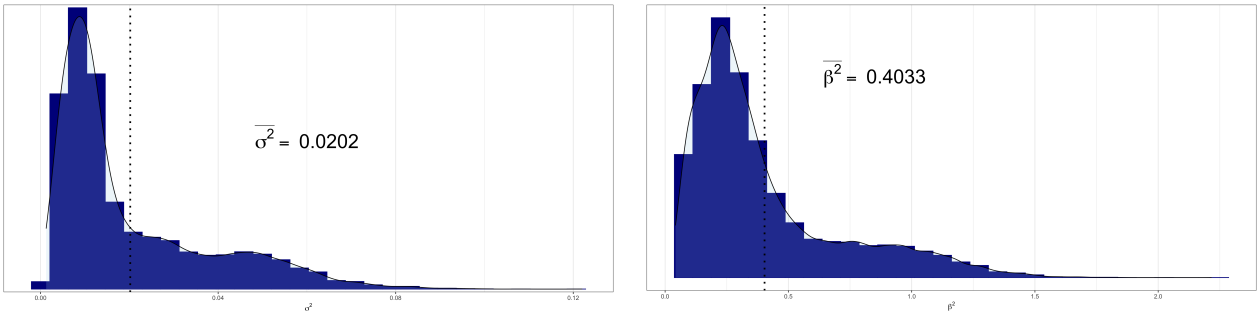
The study starts with initial parameters of $\sigma^2_{start} = 0.1$ and $\beta^2_{start} = 0.1$. It has been used BPF with Multinomail resampling with the initial values to obtain the initial hidden values needed for the algorithm. At each iteration the number of particles used are 100. Fig. 12 shows the plot of the marginal distributions for σ^2 and β^2.

The shape of Fig.12(a) and Fig.11(a) are similar. This means the distribution from the Particle Gibbs (PG) and the Particle Metropolis-Hastings (PMH) are similar. The PG sampler has been obtained with 20000 samples, 5000 more than the PMH sampler, while always considering 5000 samples discarded for the burn-in effect. The mean values are similar, indeed we have $\sigma^2_{PG} = 0.0202$ and $\sigma^2_{PMH} = 0.0231$. The discarding phase is important otherwise the final result will not respect the reality. On the same line, the shape of Fig. 12(b) and Fig. 11(b) are similar and the previous

par	$true$	par^2_{PG}	par^2_{PMH}	par_{PG}	par_{PMH}
σ	0.16	0.0202	0.0231	0.1421	0.1519
β	0.64	0.4033	0.4683	0.635	0.6843

Table 2: Comparison of PG and PMH

consideration are still valid in this case. PMH approximation overestimated β^2. Anyway, the result we have are $\beta^2_{PG} = 0.4033$ and $\beta^2_{PMH} = 0.4683$.



(a) σ^2 marginal distribution

(b) β^2 marginal distribution

Figure 12: Marginal distribution of PG sampler

The table shows that Particle Gibbs has been better for estimating σ since 0.1519 is closer to the true parameter value than 0.1421. On the other hand, Particle Metropolis Hastings perfomed better for estimating β. It returned 0.635 which is the most closest value to 0.64. The PG sampler was run with 20000 iterations where the first 5000 were discarded to correct the bias of the initial samples. The PMH was run with 15000 iterations with the 5000 iterations discarded too.

5 PyClone Light

5.1 Problem description

A collapsed Gibbs sampler has to be implemented in order to cluster mutations based on a Dirichlet Process Mixture Model (DPMM). Following the original publication, the probabilistic model can be described by the graphical model in Figure 13, where H is a Dirichlet Process (DP).

$$\begin{array}{ll} H_0 & = Beta(a_0, a_1) \\ H|H_0, \alpha & \sim DP(\alpha, H_0) \\ \phi^n|H & \sim H \\ b^n & \sim Bin(d^n, \phi^n) \end{array} \tag{12}$$

5.2 Forward simulator [Q12]

A forward simulator has been implemented, in order to generate synthetic data that accepts hyper parameters and the number of data points as inputs. For data generation a Poisson distribution with 1000 samples has been used: $d^n \sim Poisson(1000)$.

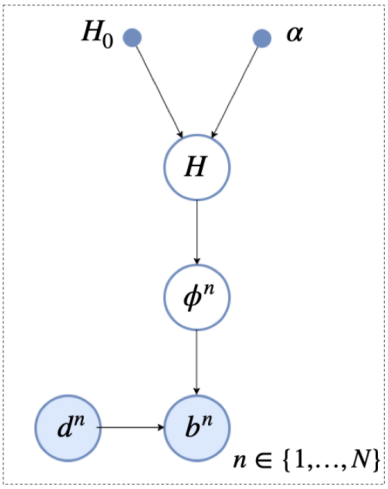


Figure 13: PyClone Light graphical model.

In the class of Infinity Mixture Models with a non-parametric prior, the Dirichlet process mixture models are a relevant family. In principle the method is working with an infinite amount of clusters which will be used to represent different distributions. This can be done with a Dirichlet process, here implemented by a stick-breaking construction. The hyper-parameter α of the Dirichlet Process (DP) and the parameters a_0 and a_1 need to be set.

Firstly a $GEM(\alpha)$ distribution is defined to generate the mixture weights classes, i.e.π. Each random variable is sampled by $Beta(\alpha)$ distribution. Using the mixture weights the component labels Z_n for the specified number of data points are drawn, associating each cluster to its mixture weight. In order to generate data the parameters d^n and ϕ^n need to be introduced: respectively the number of trials and the probability of success. For each sample $d^n \sim Poisson(d_0)$, where d_0 is set to 1000. The Beta distribution id defined by hyper parameters a_0 and a_1, same for each weight π_i. Eventually b^n is drawn from $Binomial(d^n, \phi^{Z_n})$. As said before in order to generate synthetic data stick-breaking method has been implemented, defining the probabilities of picking each individual cluster. The generated data can be visualised by a scatter plot in the d^n–b^n space.

Once fixed the random seed, several tests has been performed and presented in the next section. Some input parameters need to be defined in order to initialize the generator:

- α: hyper-parameter of DP.
- a_i: hyper-parameters of $Beta$ distribution.
- $d0$: parameter of the Poisson distribution.
- s: number of points in the d^n–b^n space.
- it: number of iterations to get convergence of the collapsed Gibbs sampler

An example is shown in Fig.14 using the settings $\alpha = 1$, $a_0 = 1$, $a_1 = 1$, $s = 100$ and $d_0 = 1000$. The majority of samples belongs to the orange group since it has a larger mixture weight, taking a larger portion of the probability.

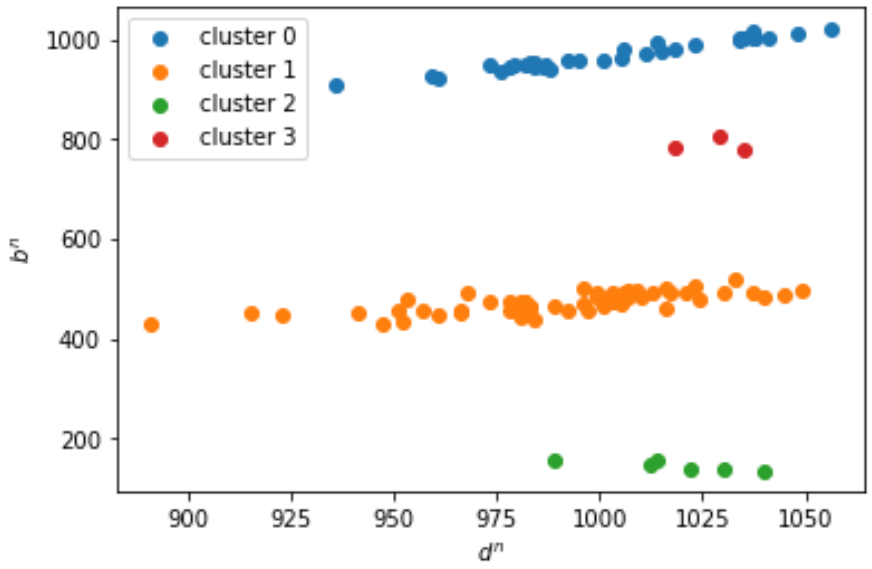


Figure 14: Forward data simulator to generate synthetic data with $\alpha = 1$, $a_0 = 1$, $a_1 = 1$, $s = 100$ and $d_0 = 1000$.

5.3 Test with synthetic data [Q13]

Since the data set available on the Github repository of the course is not labeled, a synthetic and labeled set of points has been generated in order to test the code. In order to initialize the clusters the Chinese restaurant process has been implemented. Then the Collapsed Gibbs Sampler for DPMM has been performed. For each sample the posterior predictive probability (for each of the already known clusters) and the the prior predictive (for the sample to belong to a new cluster) have been computed. Since the *Beta* and *Binomoal* distributions are conjugate, the posterior predictive of a sample (to belong to a known cluster) can be easily evaluated.

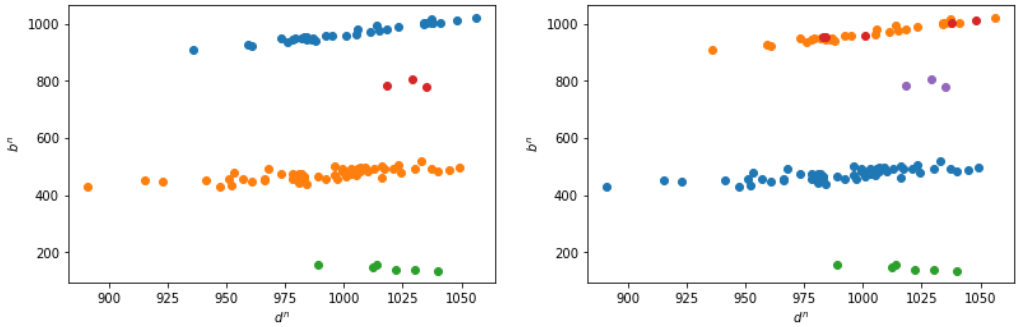
In the code some useful variables have been introduced: r_{sum} and n_{sum} are respectively the sum of all b^n and d^n belonging to a cluster a k. When the *Beta–Binomial* probability has been computed the logarithm of the probability mass function has been taken in order to avoid numerical issues, i.e. NaN values.

The figure 15 shows a very high lever of accuracy (which is defined and measured properly in the next stages). The level of agreement between the generated synthetic data and collapsed Gibbs sampler is visibly excellent.

5.4 Hyper parameters dependence [Q14]

In the d^n–b^n space the forward simulator to generate synthetic data and collapsed Gibbs sampler have been implemented, accepting hyper parameters and the number of data points as inputs. In this section the influence of the hyper parameters on final clustering have been evaluated, especially as α allows to control the number of clusters. The accuracy is measured by rand–index which is defined in the following section.

As α increases the *Beta* distribution is more and more skewed, generating values more close to zero. As a consequence smaller probabilities will be assigned to each successive mixture length w.r.t. less skewed distribution. This provides more mixtures with a significant weights, generating a larger number of cluster. Previously Fig.15 shows as with these parameters $a_0 = 1$, $a_1 = 1$, $s = 100$,

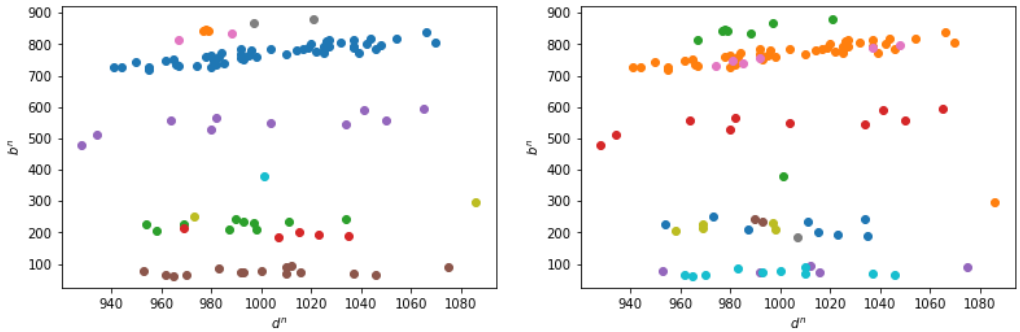


(a) Synthetic generated data with $\alpha = 1$, $a_0 = 1$, $a_1 = 1$, $s = 100$, $it = 10$ and $d_0 = 1000$.

(b) Clustering definition with with $\alpha = 1$, $a_0 = 1$, $a_1 = 1$, $s = 100$, $it = 10$ and $d_0 = 1000$.

Figure 15: The figure compares the initial clusters generated by our forward simulator and the classification carried out by our collapsed Gibbs sampler. The majority of the points are labeled correctly, only few points have been clustered wrongly. The accuracy is measured using the Rand–index (see section below) and gives a value close to 1 ($rand_i = 0.9717$).

$it = 10$, $d_0 = 1000$ and $\alpha = 1$ only 4 clusters were generated, with and accuracy $rand_i = 0.97171$. Considering $\alpha = 4$, we may observe as the number of clusters increases significantly to 10 and the accuracy decreases ($rand_i = 0.9220$). By the way the clustering algorithm still works fine.

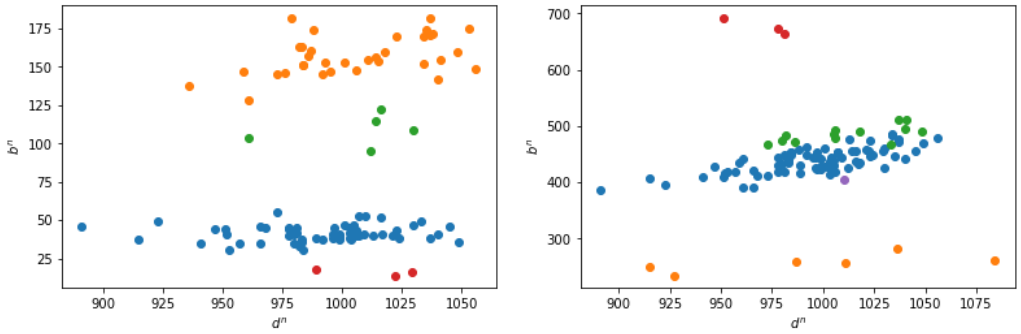


(a) Synthetic generated data.

(b) Clustering definition.

Figure 16: The figure compares the initial clusters generated by our forward simulator and the classification carried out by our collapsed Gibbs sampler. The set parameters are $\boldsymbol{\alpha = 4}$, $a_0 = 1$, $a_1 = 1$, $s = 100$, $it = 10$ and $d_0 = 1000$.

Let observe now the influence of the *Beta* parameters a_i on the data generation and the algorithm. The usual set of parameters is considered (see Fig.15). Two test cases have been considered: a first one with a anti-symmetric configuration ($a_0 = 1$, $a_1 = 10$) and a second symmetric one but with larger values w.r.t. the reference case $a_0 = 10$, $a_1 = 10$). As expected the anti-symmetric configuration allows to increase the accuracy very close to 1 ($rand_i = 0.9923$) with a clustering configuration different w.r.t. the reference case. While with larger but symmetric a_i the algorithm starts to decrease its accuracy ($rand_i = 0.7056$).



(a) Clustering definition ($\boldsymbol{a_0 = 1}$, $\boldsymbol{a_1 = 10}$). (b) Clustering definition ($\boldsymbol{a_0 = 10}$, $\boldsymbol{a_1 = 10}$).

Figure 17: The figure compares the classification carried out by our collapsed Gibbs sampler for different a_i values. The other parameters are $\alpha = 1$, $s = 100$, $it = 10$ and $d_0 = 1000$.

5.5 Accuracy and Performances analysis [Q14]

In order to evaluate the accuracy of the results the Rand–index has been used. Already introduced above, let define it properly:

$$rand_i = \frac{a+b}{\binom{N}{2}} \in [0, 1] \tag{13}$$

where a is the number of equally labelled sample couples which were also generated by the same mixtures, b is the number of differently clustered couples which were generated by the different mixtures and N is the total number of samples. It measures if each sample is exactly in the proper one, since the component label number might be different for a certain cluster and the number of clusters that the model computes can be different as well.

The accuracy decreases as α increases. Fig.18 shows an accuracy test performed with the parameters $a_0 = 1$, $a_1 = 1$, $s = 100$, $it = 10$, $d_0 = 1000$ and for α values equal to $[1, 2, 3, 5, 7, 10, 15, 18, 20]$. How α affects the accuracy is clearly visible, but the algorithm still guarantees a $rand_i$ close to 1.

On the other hand even the number of data points s can have an impact on the algorithm performances. The setting parameters are $\alpha = 1$, $a_0 = 1$, $a_1 = 1$, $it = 10$, $d_0 = 1000$ and the number of data points is varied $s = [50, 100, 150, 250, 350, 450, 600, 700, 850, 1000]$. Fig.19 shows as the accuracy remains almost constant up to $s = 400$ data points. But then it starts to decrease and close to $s = 1000$ the rand index is around $rand_i \sim 0.7$

Eventually even the number of iterations performed during the sampling by the collapsed Gibbs (to guarantee convergence) is taken into account. For the setting described above a range of different number of iterations is taken tested ($it = [5, 10, 20, 30, 40, 50, 100]$). Obviously a larger number of iterations causes a computational effort which becomes more and more relevant, but also assures an increase in the accuracy. This can be relevant especially when the performances are negatively affected by other parameters, e.g. α or s. In this particular configuration, Fig.20 shows that makes no sense considering it larger than 40/50, since no significant improvement is obtained.

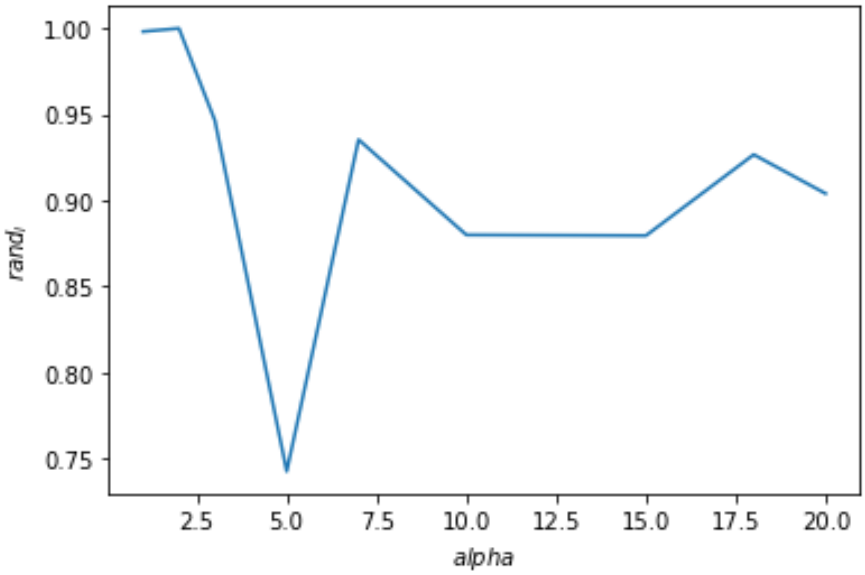


Figure 18: The accuracy is measured by $rand_i$ and the figure shows how depends from α hyper parameter.

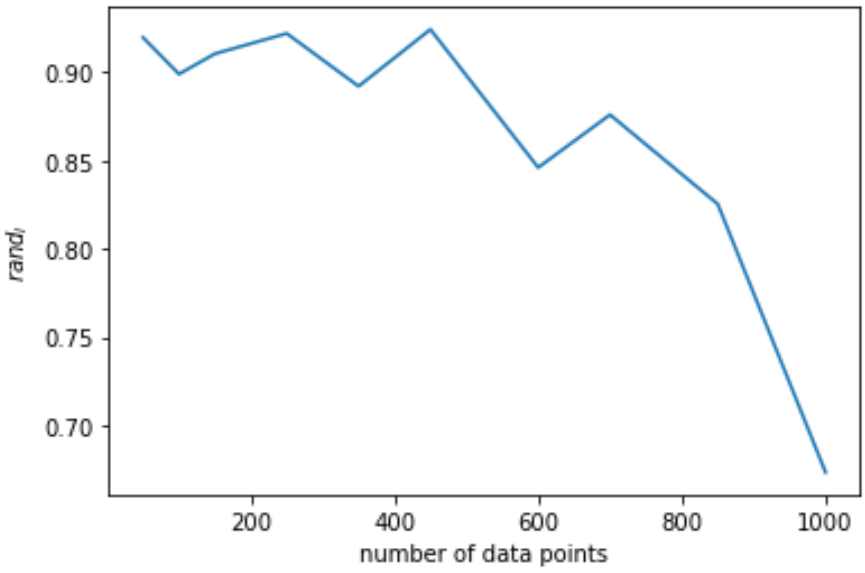


Figure 19: The accuracy is measured by $rand_i$ and the figure shows how much is affected from α the number of data points.

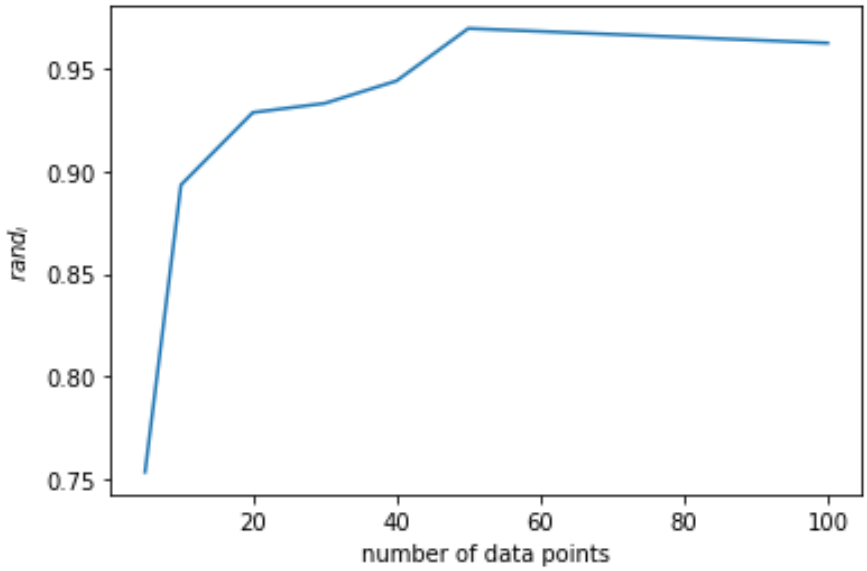


Figure 20: The accuracy is measured by $rand_i$ and the figure shows how much is affected by the number of iterations it performed by the collapsed Gibbs sampler for DPMM.