Adil SALIM Ensae 2^e Année

Méthodes séquentielles de traitement de données

University College Dublin
The Insight Centre for Data Analytics

Florian Maire
23 Juin 2014 - 31 Août 2014

Chapter 1

Note de synthèse

In this work I was interested in data analysis. I first studied how the performance of an algorithm of data analysis could be improved by ranking the data before processing. Data are values in $\mathbf{R}^d$. In a bayesian framework, the data are assumed to be random variables with a distribution parametrized by a random vector. The aim of the algorithm I studied is to infer the posterior distribution of this parameter given the data. The number of observations is assumed to be massive and therefore the algorithm is required to process a small number of data at each iteration.

I tried different way to rank the data. When the observations are one dimensional, I first measured the performance of the algorithm without ranking them *i.e* drawing them uniformly without replacement. Then, I ranked them by score of "likelihood", using a frequentist approach. An observation is attached to a high likelihood if there is a large number of observations close to it. This undermeans that the observation is in a high density region. I processed the data by increasing order with this criterion. Finally I divided the sample of data by quantiles. I made batchs with one observation in each quantile and process the data. When the observations are two dimensional, I tried two methods. In the first I did not ranked the data and in the second, I projected the data, regarded as a cloud of points, on the axis of $\mathbf{R}^d$ the closer to the cloud of points. Then I divided the projected points in quantiles. I processed the data by making batchs with observations which projection is in each quantiles, similarly to the last mentioned approach.

I measured the performance of the ranking by estimating the distance between theoretical and empirical values of some functional of the simulated samples. For one dimensional data, as expected, the worst method is the one where data are ranked by likelihood. And the best method is the one where the batchs are balanced, that is, the method where batchs are made with one observation in each quantiles. This confirm the intuition that an online inference method is better when each batch of data is representative of the full dataset. This method is significantly better than the non-ranking method. The worst method gives bad results and should not be used by statisticians. It shows that the ranking has an impact of the inference. Still, because the ranking algorithm is not computationally demanding, it does not increase the simulation time. For two dimensional data, ranking the data is slightly better than no ranking. Situations where it is really beneficial to rank the data is highly problem specific.

The next step is to try these ranking methods in very high dimensional data contexts.

I was also interested in another type of inference algorithms, used when the order of the observations is imposed. For example in Hidden Markov models, each observation only depends on the previous one through an unobserved Markov chain. An online algorithm is an algorithm which forgets each data after processing. Online algorithms are a way to save time of computation and memory in the computer. Therefore they are useful nowadays because algorithms have to process a lot of, possibly high dimensional, observations.

I tried to generalize an online algorithm which would infer a parameter of an Hidden Markov model. I generalized it to the case where the space of the observations is bigger. But it implies to do some approximations. I tried to adapt an idea of my supervisor in this case. Actually, such an algorithm seems to require too many approximations in order to be considered as an implementable solution.

Le sujet de ce stage porte sur l'analyse de données. Dans une première partie, j'ai étudié l'influence de l'ordre dans lequel les données sont traitées et son influence sur les performances d'un algorithme. Les données sont à valeurs dans $\mathbf{R}^d$. Dans le cadre d'un modèle bayésien, les données sont des variables aléatoires de loi paramétrée par un vecteur aléatoire.Le but de l'algorithme que j'ai étudié est de simuler selon la loi *a posteriori* du vecteur aléatoire c'est-à-dire de simuler selon la loi du vecteur aléatoire sachant les données. Le nombre de données est supposé être très grand donc l'algorithme doit traiter un petit nombre de données à chaque étape.

J'ai testé différentes manières de classer les données avant le traitement. Lorsque les données sont de dimension un, j'ai commencé par mesurer la performance de l'algorithme sans ranger les données c'est-à-dire en les uniformément tirant sans remise dans la masse de données. Ensuite j'ai testé de les classer par "vraisemblance" croissante, dans une approche fréquentiste. Une observation aura une vraisemblance élevée si elle est proche d'un grand nombre de données. Cela veut dire qu'elle est dans une zone de forte densité. En dernier lieu j'ai testé de ranger les données en séparant la masse de données en quantiles et en formant des paquets de données en prenant une observation dans chaque quantile. J'ai ensuite traité les paquets indépendamment. Lorsque les données sont à valeurs dans un espace de dimension deux, j'ai essayé deux méthodes. Dans la première je n'ai pas classé les données, comme précédemment. Dans la seconde, j'ai projeté les données sur l'axe le plus proche du nuage de données. Sur l'axe je les ai séparées en quantiles et comme précédemment. J'ai ensuite classée les données en utilisant les données projetées.

J'ai ensuite mesuré la performance du classement en estimant les différences entre des valeurs théoriques et des valeurs expérimentales de certaines fonctions de la simulation. Dans le cas des données unidimensionnelles, comme prévu, la méthode la moins efficace est de ranger les données par vraisemblance. La meilleure méthode est celle dans laquelle les paquets de données sont toujours équilibrés, c'est-à-dire celle où les paquets sont formés d'observations de différents quantiles. Cela appuye la conviction que le traitement séquentielle de données est meilleur lorsque chaque paquet de données est représentatif de l'échantillon de données. Cette méthode est signficativement plus performante que la méthode qui consiste à ne pas classer les données. La méthode la moins bonne donne de mauvais résultats et ne devrait pas être utilisée par les statisticiens. Cela elle montre que l'ordre dans lequel les données sont traitées a un impact sur l'estimation finale. De plus, la meilleur méthode de classement n'est pas coûteuse c'est-à-dire qu'elle n'augmente pas la durée d'une simulation. Dans le cas des données bidimensionnelles, le classement (réèl) des données donne un résultat légèrement meilleur. Les situations dans lesquelles l'on peut tirer d'un classement dépend fortement du problème considéré. La prochaine étape est de tester ce classement en grande dimension. Mais les résultats en grande dimension ne devrait pas être meilleurs qu'en dimension deux.

Dans une deuxième partie de ce stage, j'ai étudié un autre type d'algorithme d'inférence, qui sont utilisables dans un cas où les données ont un ordre prédéfini. Par exemple, dans les modèles de Markov cachés, chaque observation dépend d'une certaine manière de la précédente, à travers une chaîne de Markov non observable. Certains algorithmes traîtent les données en ligne, c'est-à-dire en oubliant chaque donnée après son traitement. Ce genre d'algorithme permettent d'économiser du temps de calcul et de la mémoire. Ils sont donc très utiles de nos jours, à l'heure où les masses de données à traîter sont de plus en plus grandes.

J'ai essayé de généraliser un algorithme qui traîte un modèle de Markov caché en ligne afin d'inférer un paramètre de ce modèle de Markov caché. J'ai pensé une généralisation possible dans le cas où la chaine de Markov sous-jacente prend ses valeurs dans un espace possiblement très grand (non dénombrable). Cela implique d'effectuer certaines approximations. J'ai utilisé une idée que mon maître de stage a développé dans un autre cadre. Mais dans le cas présent, le nombre d'approximations à effectuer est trop grand pour que la généralisation de cet algorithme soit implémentée.

Contents

Chapter 2

Introduction

I am really grateful to Florian Maire for these ten weeks in Dublin, for supervising my work, and for the freedom he gave me in my work. I also want to thank Pierre Alquier for his advices and Nial Friel for welcoming me in the Insight Centre for Data Analytics.

Data analysis is about processing observed data in order to extract information from them. The observations can be dependant or independant and can be real numbers or vectors. Improvement of computer performances have allowed statisticians to develop many methods to process data. But today, because of the large amont of data available, we need algorithms which can be applied to a lot of observations. One of the approach is to process them *online*, that is to process the observations one by one or by small (non overlapping) batchs and to forget them forever after processing. Such methods allow the computer not to accumulate observations, which can lead to a significant save in memory, especially when data are high dimensional vectors. Subsequently for iterative algorithms, the *online* paradigm allows not to browse all the data at each iteration, which can considerably speed up the inference.

In the first part of my work, I have investigated if, in situations where the observations are independant and identically distributed, the order in which we process them has any kind of influence on the inference quality. To address this question, the milestones I defined were twofold: (i) find an algorithm which processes the data online and (ii) compare the influence of different ranking strategies on the main algorithm efficiency. The Iterated Batch Importance Sampling (IBIS) algorithm (see [1]), which takes inspiration from the Sequential Importance Sampling with Resampling methodology but for static models, matches our framework and was used as an example of *online* algorithm. Different heuristics were considered when dealing with the ranking of the data. For one dimensional simulated data, we see that some stragies of ranking are really better than others. For example we can divide the bias by three with a good strategy of ranking. It can also reduce the variance of this bias on several simulations. In dimension two, the results are not as striking but a proper ranking could still yield a slight impact. Therefore ranking methods can be used to improve the inference.

On the other hand, I was also interested in algorithms for dependant data which yields a totally different approach: indeed, in such situations one cannot rank the data before using them. Still, some existing online algorithms have been proposed to infer typically non independent data such as Hidden Markov models (a model featuring markovian dependence between the observations), see [5]. We have questionned the feasibility of extending such an approach to more complex models, where typically the conditional expectation that appears at each iteration of the algorithm cannot be computed in closed form, following the development of [6].

The report is organized as follows. In a first part we will present an algorithm which process independant data, the IBIS algorithm and try different way to rank the data. In a second part we will try to adapt the Stochastic Approximation online Expectation-Maximization (SAoEM) algorithm, an algorithm which designed for independant data, to the case where data are dependant and following an Hidden Markov model.

Chapter 3

Is there a way to rank the data?

3.1 Importance sampling

Let (Θ, B, λ) be a measured space. Let $p(\mathrm{d}\theta)$ and $q(\mathrm{d}\theta)$ be two probability measures on this space with density p and q according to λ.

Assume that the support of p is included in the support of q. For any measurable and $p(\mathrm{d}\theta)$-integrable real function f our aim is to approximate

$$\int_\Theta f(\theta)\, p(\mathrm{d}\theta) = \int_\Theta f(\theta)p(\theta)\, \lambda(\mathrm{d}\theta) = \int_\Theta f(\theta)\frac{p(\theta)}{q(\theta)}\, q(\mathrm{d}\theta)$$

where all the quantities written are well defined. If sampling from q is feasible, one can sample $\theta_1, \theta_2, \ldots, \theta_n$ a sequence of independent and identically distributed random variables from q. According to the strong law of large numbers,

$$\frac{1}{n}\sum_{i=1}^{n} f(\theta_i)\frac{p(\theta_i)}{q(\theta_i)} \longrightarrow \int_\Theta f(\theta)\, p(\mathrm{d}\theta)$$

and

$$\frac{1}{n}\sum_{i=1}^{n} \frac{p(\theta_i)}{q(\theta_i)} \longrightarrow \int_\Theta p(\mathrm{d}\theta) = 1$$

a.s. when n grows to infinity.

Therefore,

$$\frac{\sum_{i=1}^{n} f(\theta_i)\frac{p(\theta_i)}{q(\theta_i)}}{\sum_{i=1}^{n} \frac{p(\theta_i)}{q(\theta_i)}} \longrightarrow \int_\Theta f(\theta)\, p(\mathrm{d}\theta)$$

a.s.

Notation : $\forall i \leq n, w_i = \frac{p(\theta_i)}{q(\theta_i)}$**. The sequence** $((\theta_i, w_i))_i$ **is called a particle filter targeting** p**. The** θ_i **are the particles and the** w_i **are the unormalized importance weights.**

Assume that n is fixed. The two following points justify using of the IBIS algorithm.

- We have a specific particle filter and we want to compute an expectation under an other probability measure, say $r(d\theta)$ which admits a density r with respect to λ. To do so, we only need to update the weights

$$\forall i, w_i \longleftarrow w_i \frac{r(\theta_i)}{p(\theta_i)}.$$

  By straightforawrd algebra, it can be shown that for a general particle filter $((\theta_i, w_i))_i$ targeting p, the reweighting step $\forall i, w_i \longleftarrow w_i \frac{r(\theta_i)}{p(\theta_i)}$ results in a particle filter targeting r ; an idea which has been intensively used to design IBIS ([1]).

- If $((\theta_i, w_i))_i$ is targeting a distribution p, and if $\lambda > 0$ is a real number, independant of i, then $((\theta_i, \lambda w_i))_i$ is also targeting p because the importance sampling estimator remains unchanged

$$\frac{\sum_{i=1}^n \lambda w_i f(\theta_i)}{\sum_{i=1}^n \lambda w_i} = \frac{\sum_{i=1}^n w_i f(\theta_i)}{\sum_{i=1}^n w_i} \longrightarrow \int_\Theta f(\theta)\, p(\mathrm{d}\theta).$$

3.2 Iterated Batch Importance Sampling (IBIS) [1]

Let N be a fixed positive integer and $Y_1, Y_2, \dots, Y_N$ be a sequence of independent and identically distributed random variables. We use the Lebesgue measure as the reference measure. Assume that

$$\forall i, Y_i \sim p(.|\theta)$$

where p is the likelihood of one observation, and the distribution of the Y_i is parameterized by a random vector θ.

- We use a bayesian strategy : the prior distribution of θ is $\pi(.)$ and the posterior distribution of interest is $\pi(.|Y_1, Y_2, \dots, Y_N)$.
- The idea of IBIS is to compute a particle filter targeting $\pi(.|Y_1, Y_2, \dots, Y_{k+q})$ for the current k given a particle filter targeting $\pi(.|Y_1, Y_2, \dots, Y_k)$. At the end of the algorithm we get a particle filter targeting $\pi(.|Y_1, Y_2, \dots, Y_N)$.
- One step of the algorithm is the reweighting:

$$w_i \leftarrow w_i \times \frac{\pi(\theta_i|Y_1, Y_2, \dots, Y_{k+q})}{\pi(\theta_i|Y_1, Y_2, \dots, Y_k)} \propto w_i \times p(Y_{k+1}|\theta_i)...p(Y_{k+q}|\theta_i), \forall i.$$

  Where $\propto$ means proportionnal up to the constant

$$\frac{\int p(Y_1|\theta)...p(Y_k|\theta)\pi(\theta)\,\mathrm{d}\theta}{\int p(Y_1|\theta)...p(Y_{k+q}|\theta)\pi(\theta)\,\mathrm{d}\theta}$$

  where $\pi(\mathrm{d}\theta)$ denotes the prior distribution on the parameters. This constant does not depend on θ_i. In this algorithm, the data are sliced into batches of the same size, say q. That is, at the first step of the IBIS algorithm the particle filter targets $\pi(.|Y_1, Y_2, \dots, Y_q)$, at the next step it targets $\pi(.|Y_1, Y_2, \dots, Y_{2q})$, at the next step it targets $\pi(.|Y_1, Y_2, \dots, Y_{3q})$ etc. This algorithm is an online algorithm because if the particle filter is currently targeting $\pi(.|Y_1, Y_2, \dots, Y_k)$, for the reweighting step

$$w_i \leftarrow w_i \times \frac{\pi(\theta_i|Y_1, Y_2, \dots, Y_{k+q})}{\pi(\theta_i|Y_1, Y_2, \dots, Y_k)} \propto w_i \times p(Y_{k+1}|\theta_i)...p(Y_{k+q}|\theta_i), \forall i,$$

we only need to know the next batch $Y_{k+1}, ..., Y_{k+q}$ and we don't need to remember the past observations $Y_1, ..., Y_k$. A convergence result of a measure of efficiency of the method was shown in [1] for q growing at each step but we will conserve a fixed number of observations by batch.

3.2.1 Degeneracy $\longrightarrow$ Resampling

As described, the algorithm is likely to suffer from degeneracy: some particles will get almost all the weight of the particle filter and other particles will have a small weight (close to zero) because at each step we are multiplying the weights. It's the weight degeneracy. The particles with a small weight are not useful for the particle filter (to target the right distribution) but they add variance to the particle filter. To avoid it, and as suggested in [1], we will resample the particle filter, that is duplicate particles with an high weight and forget particles with a small one and put all the weights to one.

I used a multinomial selection. If the normalised weight of θ_i is $w'i = \frac{w_i}{\sum_{i=1}^{n} w_i}$. I drew the random vector $(a_1, ..., a_n)$ from a multinomial distribution of parameters $(n, (w'_i)_{i \in [0,n]})$. Then I duplicated the particle θ_i a_i times.

After the resampling step, we have a particle filter of n particles, the same than before the resampling, but some particles have been duplicated and some have been deleted. And the new unormalized weights are all one. It avoids very large weights, and by getting rid of the particles which are not useful for the inference (with a small weight), it saves some time of calculation. But it doesn't avoid the degeneracy as well. After the resampling the particle filter is still targetting $\pi(.|Y_1, ..., Y_{k+q})$.

3.2.2 High correlations $\longrightarrow$ Move with a markov chain kernel

To duplicate some particles introduces high correlations between the particles of the particle filter. To avoid these correlations in the particle filter we add a move step as follows.

Assume the particle filter is now targeting the distribution $\pi(\theta|Y_1, Y_2, \dots, Y_k)$ and we have reweighted and resampled the particle filter. Assume we have a Markov kernel Q which has $\pi(.|Y_1, Y_2, \dots, Y_k)$ as its unique stationnary distribution. If we apply *one time* the Markov kernel Q to *all* the particles, we will get new particles sampled from $\pi(\theta|Y_1, Y_2, \dots, Y_k)$ and it will reduce the correlations in the particle filter because it will move the particles. For example, if θ_i and θ_j where equal before the move step, they were highly correlated, but after the move step they are generally no longer equal.

3.2.3 Fast mixing kernel : one application to all particles

In this approach we intend to apply one time a fast mixing Markov kernel Q to the particles to reduce the correlations.

When the observations are real numbers, I tried a Metropolis-Hastings kernel. I used the Metropolis-Hastings algorithm in the independant case with a normal distribution f as a proposal distribution. The expectation of the normal distribution is the empirical mean of the particles and the variance the empirical variance of the particles. I applied one step of the Metropolis-Hastings algorithm to each particle θ which can be discribed as follows. Given the current particle filter $((\theta_i, 1))_{i\in[0,n]}$ (after the resampling the unormalized weights are all one), compute the empirical mean and the empirical variance of the particle filter : $\mu = \frac{1}{n}\sum_{i=1}^{n}\theta_i$ and $v = \frac{1}{n}\sum_{i=1}^{n}(\theta_i - \mu)^2$. Let h be the density of the normal distribution $N(\mu, \sqrt{v})$ of mean μ and variance v. Then, for all $i \in [0, n]$,

- Draw $t \sim N(e, v)$
- Draw $u \sim U(0, 1)$
- Compute
$$\alpha(\theta_i, t) = \frac{h(\theta_i)\pi(t)\prod_{j=1}^{k+q} p(Y_j|t)}{h(t)\pi(\theta)\prod_{j=1}^{k+q} p(Y_j|\theta)}$$
(At this step we browse all the observations)
- If $u < \alpha(\theta_i, t)$ then $\theta_i \longleftarrow t$

This is the move step. I used the Metropolis-Hastings algorithm in an independant case. The move step is one application to all particles of the reversible Markov kernel

$$Q : (\theta, A) \longmapsto \int_A h(t)\alpha(\theta, t)\mathrm{d}t + 1_A(\theta)\int_{\mathbf{R}} h(t)(1 - \alpha(\theta, t))\mathrm{d}t$$

for all $\theta \in \mathbf{R}$ and all $A \subset \mathbf{R}$ measurable.

Looking for a fast mixing kernel, I tried to use a simulation method I studied last year. Its name is the Hamiltonian Monte Carlo algorithm. It looks like the Monte Carlo algorithm but at each iteration of the Hamiltonian Monte Carlo algorithm one have to solve (numerically) a differential equation. To sample for a distribution called the banana, this algorithm converges faster than the Metropolis-Hastings algorithm. Let L be the intermediate number of step we use to solve the differential equation we solve at each iteration of the Hamiltonian Monte Carlo. In one iteration of Metropolis Hastings we browse one time all the data $Y_1, ..., Y_N$ when we compute the ratio $\alpha(\theta, t)$. In one iteration of Hamiltonian Monte Carlo we browse L times all the data. At each step of the integration of the differential equation we have to compute the gradient of

$$\theta \longmapsto \log(\pi(\theta|Y_1, Y_2, \ldots, Y_{k+q}))$$

in one point so in the Hamiltonian Monte Carlo algorithm we need to browse the data L times. Moreover it can fail if large numbers appear while computing the gradient of this function.

3.2.4 An approximate move with nonparametric estimation?

Because of the move step we no longer work online. To apply the Markov kernel Q we have to browse all the past observations $Y_1, ..., Y_k$. To solve this problem I thought about getting an approximation of $\pi(\theta|Y_1, Y_2, \ldots, Y_k)$ using the particles after the resampling step and apply the move step with a kernel which have the distribution $\widehat{\pi_k}(\theta)$ (an approximation of $\pi(\theta|Y_1, Y_2, \ldots, Y_k)$) as its unique stationnary distribution. $\widehat{\pi_k}(\theta)$ is a kernel estimator of the density of $Y_1, ..., Y_k$. If $Y_1, Y_2, \ldots, Y_k$ are drawn from $\pi(\theta|Y_1, Y_2, \ldots, Y_k)$, using the nonparametric statistics theory we know that the error between $\pi(\theta|Y_1, Y_2, \ldots, Y_k)$ and its approximation $\widehat{\pi_k}(\theta)$ decreases to 0 as the number of observations k grows to infinity.

Assume that $Y_1, Y_2, \ldots, Y_k$ are drawn from $\pi(\theta|Y_1, Y_2, \ldots, Y_k)$ and we have an nonparametric estimation $\widehat{\pi_k}(\theta)$ of $\pi(\theta|Y_1, Y_2, \ldots, Y_k)$ using the particles after the resampling step. We move the particles with a Markov kernel $\widehat{Q}$ which has $\widehat{\pi_k}(\theta)$ as its unique stationnary distribution. Before the move step the particles are not really drawn from $\widehat{\pi_k}(\theta)$ (it is an estimation of the density), so after the move step, we get particles which are not really drawn from $\widehat{\pi_k}(\theta)$ but we don't take into account this fact (for an ergodic Markov kernel $\widehat{Q}$ the distribution of the particles are closer to $\widehat{\pi_k}(\theta)$ after the move step than before). After the move step with $\widehat{Q}$, we have particles sampled from $\widehat{\pi_k}(\theta)$ and the particle filter targets $\widehat{\pi_k}(\theta)$. At the next step of the IBIS algorithm, we keep the reweighting step so the new particle filter won't target $\pi(\theta|Y_1, Y_2, \ldots, Y_{k+q})$ because the current particle filter is targeting $\widehat{\pi_k}(\theta)$ rather than $\pi(\theta|Y_1, Y_2, \ldots, Y_k)$. And the next move step will introduce a new error of sampling etc. There is no obvious reason for the error of simulation to decrease. In nonparametric statistics theory we have asymptotical results for the kernel estimator of a fixed density when the number of observations grows to infinity, whereas in the IBIS algorithm, the target distribution changes a each step. Moreover, if we imagine that we only have one step in the IBIS algorithm which takes the batch of all the N observations, at the end of the algorithm the particle filter won't target $\pi(\theta|Y_1, Y_2, \ldots, Y_N)$ but $\widehat{\pi_N}(\theta)$ (or a distribution closed to this one).

Actually, there is "no free lunch". Using only the information contained in the particle filter, we cannot remove the correlations between the particles without loosing something. Otherwise, it means that we won information about the particles for "free", knowing nothing but the information in the current particles .

3.2.5 Evaluation of the results

At the end of IBIS we have a particle filter where the weights are all equal to one and the particles are sampled from $\pi(\theta|Y_1, Y_2, \ldots, Y_N)$. To measure the quality of the simulation we compute, for some integrable function f,

$$|\frac{1}{n}\sum_{i=1}^{n} f(\theta_i) - \int_{\Theta} f(\theta)\, \pi(\mathrm{d}\theta|Y_1, Y_2, \ldots, Y_N)|.$$

For example, I used $f : x \longmapsto x$ and $f : x \longmapsto x^2$ in my simulations in dimension one.

We can also use the total variation distance to measure the distance between the empirical distribution

$$\frac{1}{n}\sum_{i=1}^{n}\delta_{\theta_i}$$

(where the θ_i are the particles at the end of the algorithm and δ_x is the Dirac measure in x) and the theoretical one

$$\pi(\mathrm{d}\theta|Y_1, Y_2, \ldots, Y_N).$$

The empirical distribution does not have density according to Lebesgue measure so we connot use simple formulas to compute the total variation distance between the two distributions.

3.3 How to rank the data in dimension one

3.3.1 Data

Data are sampled from a mixture of two normal distributions p,

$$p(x) = 0.6p_{0,1}(x) + 0.4p_{2,0.1}(x)$$

where $p_{m,\sigma}$ is the density of a normal distribution $N(m, \sigma)$ of mean m and standard deviation σ, see figure 3.1.

The prior distribution π of the particles is a normal distribution $N(2, 1)$. At the first iteration of the IBIS algorithm, the first reweighting is

$$w_i \leftarrow w_i \times \frac{\pi(\theta_i|Y_1, Y_2, \ldots, Y_q)}{\pi(\theta_i)} \propto w_i \times p(Y_{k+1}|\theta_i)...p(Y_q|\theta_i), \forall i.$$

At the initialization, the particles are sampled from $\pi = p_{2,1}$. The likelihood of the observations is $p_{\theta,1} = p(.|\theta)$ the density of the distribution $N(\theta, 1)$. And one can derive the posterior distribution of the particles

$$\pi(\theta)\prod_{i=1}^{N} p(y_i|\theta) \propto \exp(-\frac{1}{2}((N+1)\theta^2 - (4 + 2\sum_{i=1}^{N} y_i)\theta + (4 + \sum_{i=1}^{N} y_i^2))).$$

It's a normal distribution $N(m, \sigma)$ of mean

$$m = \frac{4 + 2\sum_{i=1}^{N} y_i}{2N + 2}$$

and variance

$$\sigma^2 = \frac{1}{N+1}.$$

The second moment of this distribution is

$$m^2 + \sigma^2 = \left(\frac{4 + 2\sum_{i=1}^{N} y_i}{2N + 2}\right)^2 + \frac{1}{N+1}.$$

We will need these quantities for the evaluation of the results. I used 100 observations and did batchs of size 5. The number of particles in the particle filter is 100.

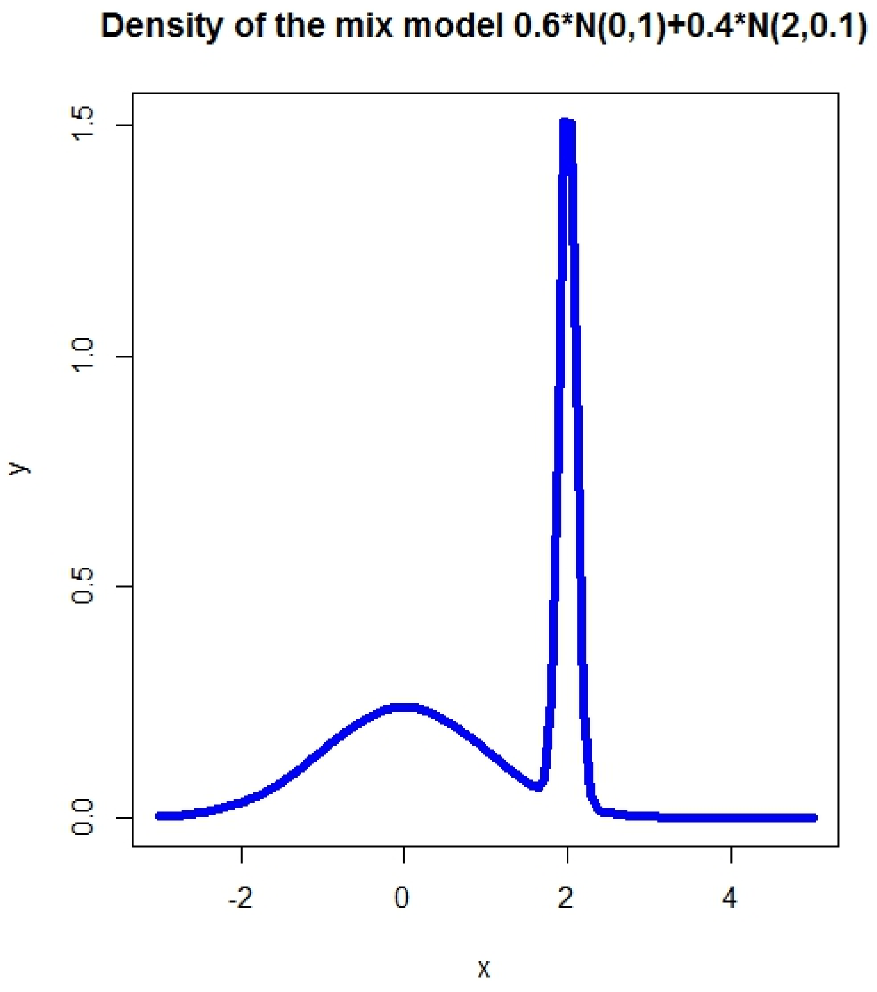

Figure 3.1: Mixture model in dimension one

For each method of ranking I did 74 simulations and computed

$$|\frac{1}{n}\sum_{i=1}^{n}\theta_i - \int_\Theta \theta\,\pi(\mathrm{d}\theta|Y_1,Y_2,\dots,Y_N)|$$

and

$$|\frac{1}{n}\sum_{i=1}^{n}\theta_i^2 - \int_\Theta \theta^2\,\pi(\mathrm{d}\theta|Y_1,Y_2,\dots,Y_N)|$$

at each simulation. Then I computed the mean and the variance of these quantities on the 74 simulations.

3.3.2 By drawing without replacement

The first setup consists in drawing the data without replacement: this is a non-ranking strategy to which we will compare other ranking approaches. I used sampled data and divided them by batchs. It's like drawing the batchs without replacement. With this method, the batchs should be *representative* of the distribution of the observations, that is the distribution of the batchs should be close to the distribution of the whole sample of data. For example, we select with an high probability observations from an high probability region.

We consider two strategies to rank the data.

3.3.3 By score of likelihood

An other approach is to really rank the data. There are different way to do that, for example using a criterion which score the observations. An idea is to rank the data by "representativity of the sample" or by "likelihood" (I don't use the word "likehood" is its traditional sense in statistics). I wondered if there is a canonical way to score the data, the score representing their representativity in the whole sample. We assume in this section that the data $Y_1, ..., Y_N$ are points in $\mathbf{R}^d$ and use a deterministic approach. Let

$$f : (x_1, ..., x_N) \longmapsto f(x_1, ..., x_N)$$

be a map from $\mathbf{R}^{d^N}$ to $\mathbf{R}$ which represents the score of x_1. What can we say about f?

First, if we turn the point Y_i around Y_1 it shouldn't change the score. Therefore, f only depends on the distance from Y_1 to the other data points, or any other injective function on these distances. Let be $\|.\|$ the euclidean norm. We can write

$$f(x_1, ..., x_N) = g(K(\|x_1 - x_2\|), ..., K(\|x_1 - x_N\|))$$

for all choice of $(x_1, ..., x_N)$, Where

$$g : (a_2, ..., a_N) \longmapsto g(a_2, ..., a_N)$$

from $\mathbf{R}^{N-1}$ to $\mathbf{R}$ and

$$K : x \longmapsto K(x)$$

is an injective map from $\mathbf{R}_+$ to $\mathbf{R}$. Now, if x_1 is going far from the other points of the cloud x_i the score of x_1 should decrease so we choose g increasing for each a_i and K decreasing.

Then g is a symmetric function : the value of $g(a_2, ..., a_N)$ does not depend on the rank of the a_i. This essentially means that the values of g on $\mathbf{R}^{d^{N-1}}$ are determined by the values of g on $S = \{(a_2, ..., a_N) \in \mathbf{R}^{d^{N-1}}, a_2 \leq a_3 \leq ... \leq a_N\}$. But there are results from the theory of symmetric functions which explain that we can change the variable of g and conserve the regularity of the function g.
Let p be a positive integer and $\sigma_1(z_1, ..., z_p), ..., \sigma_p(z_1, ..., z_p)$ be the elementary symmetric functions in $(z_1, ..., z_p)$

$$\forall k \in \{1, ..., p\}, \sigma_k(z_1, ..., z_p) = \sum_{1 \leq s_1 < ... < s_k \leq p} \prod_{j=1}^{k} z_{s_j}.$$

In the theory of symmetric functions we know that, if

$$g : (a_2, ..., a_N) \longmapsto g(a_2, ..., a_N)$$

is a symmetric *polynomial* function, there is an unique polynomial function h and an unique function $\tilde{h}$ such that

$$g(a_2, ..., a_N) = h(\sigma_1(a_2, ..., a_N), ..., \sigma_{N-1}(a_2, ..., a_N))$$

and

$$g(a_2, ..., a_N) = \tilde{h}(\sum a_i, ..., \sum a_i^{N-1})$$

There are analogous of this theorem for other regularity : for example if g is a *smooth* function we can find an unique *smooth* function h such that the equality is right for all $(a_2, ..., a_N) \in \mathbf{R}^{N-1}$ ([7]). Therefore, if we assume that g is a smooth function, we can rewrite

$$f(x_1, ..., x_N) = h(\sigma_1(K(\|x_1 - x_2\|), ..., K(\|x_1 - x_N\|)), ..., \sigma_{N-1}(K(\|x_1 - x_2\|), ..., K(\|x_1 - x_N\|)))$$

or,

$$f(x_1, ..., x_N) = h(K(\|x_1 - x_2\|) + ... + K(\|x_1 - x_N\|), ..., K(\|x_1 - x_2\|) \times ... \times K(\|x_1 - x_N\|))$$

for all $(x_1, ..., x_N)$ where h is a smooth function. (And, in the equality if we take $\tilde{h}$ to be a linear form we find all functions

$$f(x_1, ..., x_N) = \frac{1}{N-1} \sum_{i=2}^{N} P(\|x_1 - x_i\|)$$

where P is a polynomial function of degree$< N$.)

Under this assumption, the choice of a "good" function f for our problem is the choice of K and h.

Rewriting f in this form reminds the kernel estimator : if K is a kernel and h is the first coordinate application we find a function which looks like the kernel estimator. But we made some regularity assumption so we don't find exactly all kernels. For example the kernel of the histogram estimator is not a smooth function on $\mathbf{R}_+$. But there are analogous results if we replace smooth function by continuous function. We take K as a smooth stricly decreasing function, and h as a smooth function.

In the initial problem, N is supposed to be big. Except $K(\|x_1 - x_2\|) + ... + K(\|x_1 - x_N\|)$ and $K(\|x_1 - x_2\|) \times ... \times K(\|x_1 - x_N\|)$ the other terms are the result of at least $O(N^2)$ operations. We will choose h independant on this terms. The term $K(\|x_1 - x_2\|) \times ... \times K(\|x_1 - x_N\|)$ is the product of a lot of numbers and it can give a large number or a number close to zero. When there is a product of a lot of numbers in algorithms, one often use the log to reduce the number. If we compute the log of this term, it will change the product in a sum like the first term and we will get another "kernel" $\log(K)$. We finally choose to take h only dependant on the first term, the sum $K(\|x_1 - x_2\|) + ... + K(\|x_1 - x_N\|)$.

If the point x_i of the cloud are going far from x_1, $f(x_1, ..., x_N)$ must decrease strictly so we will choose h stricly increasing. Our aim is to compare the values of $f(x_1, ..., x_N)$ to score the data in order to rank them. We are now working for x_1 but we will do the same for the other x_j and we will compare the values of the numbers $h(K(\|x_j - x_2\|) + ... + K(\|x_j - x_N\|))$ for all j to score and rank the observations. Because we are interested in ranking, we only need to know the numbers $K(\|x_i - x_2\|) + ... + K(\|x_i - x_N\|)$ to rank the data. Finally, the expression is similar to the kernel estimator so I used it directly for the simulations. I used a gaussian kernel and a bandwidth of 0.94.

Let's forget the bayesian framework in this part. Given the data $Y_1, ..., Y_N$, I did a nonparametric estimation of the density of the distribution of the data (with a gaussian kernel) and found a function h. I computed $h(Y_1), ..., h(Y_N)$ and scored the data with these numbers and ranked them in increasing order. These numbers can be interpreted as estimations of the likelihood of the distribution of the $Y_1, ..., Y_N$. I performed the IBIS algorithm with this ranking.

We expect this ranking to be very inefficient. The data are sampled from a mixture of normal distribution. At the end of the algorithm, the data processed are only close to 0 or very close to 2. Whereas at the beginning of the algorithm there are farer from 0 or 2. So at the beginning of the algorithm the particles are in a region and then we start to process "only 0 and 2" and it can trap the particles in another region.

3.3.4 By quantiles

In this third approach we force the batchs to be representative : the empirical distribution of the batchs will be closer to the distribution of the observations. The observed data are real numbers. We identify the quantiles of the sample and make the batchs by choosing (randomly) one observation in each quantile. In this method the batchs are more representative, for example we cannot have a batch where all the data are in one quantile. With this method we avoid the phenomenon of the previous section. The batchs are always balanced.

3.3.5 Results

On the 74 simulations of the three previous methods, the mean and the variance of the bias

$$|\frac{1}{n}\sum_{i=1}^{n}\theta_i - \int_{\Theta}\theta\,\pi(\mathrm{d}\theta|Y_1, Y_2, \ldots, Y_N)|$$

By drawing without replacement		By score of likelihood		By quantiles	
Mean	Variance	Mean	Variance	Mean	Variance
0.00217	0.00070	0.75573	0.05521	0.00071	0.00020
0.00300	0.00191	0.63823	0.06414	0.00090	0.00047

Table 3.1: Results in dimension one

is on the first line of the table 3.1 and the mean and the variance of

$$|\frac{1}{n}\sum_{i=1}^{n}\theta_i^2 - \int_{\Theta}\theta^2\,\pi(\mathrm{d}\theta|Y_1, Y_2, \ldots, Y_N)|$$

is on the second line of the table 3.1.

A method is undersood to be better than another one when the mean and the variance of the bias of the first and second order statistics are smaller. As expected, the best strategy is clearly to rank the data by quantiles and the worst to rank them by likelihood, the *naive* ranking liying in between. Ranking by quantiles yields a factor 2 between the two means of the quantities, compared to the drawing without replacement. And it divides the variances by almost four. Ranking the data by score of likelihood multiplies the means by 200 and it also increases the variances compared to the drawing without replacement. This strategy actually consists in feeding the algorithm with batches that are highly unrepresentative of the full dataset, hence yielding unsurprising poor results. When ranking the data by likelihood in ascendant order, the method starts infering the posterior through data that are in the tail of the likelihood. Thus, early estimates obtained from misleading observations may dramatically affect the convergence of the method. Reciprocally, when ranking the data by likelihood in descendant order, the estimate may fail to stabilize due to unconsistancy of the observations processed while having supposedly reached stationarity, hence yielding high variance in the monitored statistics.

In addition, in the three previous methods, there is no significant difference of processing time in the IBIS algorithm.

3.4 How to rank the data in dimension $d > 1$

3.4.1 Data

The new prior distribution π of the particles is $(\theta[1], \theta[2]) \longmapsto \pi(\theta[1], \theta[2]) = p_{2,20}(\theta[1])p_{2,20}(\theta[2])$. The likelihood of the observations is $p(.|(\theta[1], \theta[2])) : (y[1], y[2]) \longmapsto p_{\theta[1],20}(y[1])p_{\theta[2],20}(y[2])$. It means that the components of the particles and the observations are supposed to be independant. One can derive the posterior distribution of the particles

$$\pi(\theta)\prod_{i=1}^{N}p(y_i|\theta) = \left(p_{2,20}(\theta[1])\prod_{i=1}^{N}p_{\theta[1],20}(y[1]_i)\right)\left(p_{2,20}(\theta[2])\prod_{i=1}^{N}p_{\theta[2],20}(y[2]_i)\right)$$

$$\propto \prod_{j\in\{1,2\}}\left(\exp\left(-\frac{1}{2\times 20^2}\left((N+1)\theta[j]^2 - (4 + 2\sum_{i=1}^{N}y[j]_i)\theta[1] + (4 + \sum_{i=1}^{N}y[j]_i^2)\right)\right)\right).$$

In this expression, we can clearly see how to generalize in dimension $d > 2$. This is a two dimensional normal distribution $N(m, \sigma)$ with mean

$$m = \left(\frac{4 + 2\sum_{i=1}^{N} y[1]_i}{2N+2}, \frac{4 + 2\sum_{i=1}^{N} y[2]_i}{2N+2} \right)$$

and variance

$$\sigma^2 = diag\left(\frac{20^2}{N+1}, \frac{20^2}{N+1} \right).$$

The second moment of the first component is

$$\left(\frac{4 + 2\sum_{i=1}^{N} y[1]_i}{2N+2} \right)^2 + \frac{20^2}{N+1}$$

and we get the second moment of the second component by replacing [1] by [2]. The first crossed moment is (by independance)

$$\frac{4 + 2\sum_{i=1}^{N} y[1]_i}{2N+2} \times \frac{4 + 2\sum_{i=1}^{N} y[2]_i}{2N+2}$$

I did 100 simulations by drawing without replacement and by performing the PCA method (see the following section). There are generalizing in dimension two the two best methods in dimension one.

I computed the mean and the variance of these quantities on the 100 simulations. I used 100 observations and did batchs of size 5. The number of particles in the particle filter is 100.

I used two kinds of data. I first considered data from a mixture of two Gaussians $N((0,0), diag(20,20))$ and $N((10,10), diag(2,2))$ with the density (see the figure 3.2)

$$(y[1], y[2]) \longmapsto 0.6 p_{0,20}(y[1]) p_{0,20}(y[2]) + 0.4 p_{10,2}(y[1]) p_{10,2}(y[2]).$$

Then, I considered data from a 2-dimensional banana shape target, with density proportional to (see the figure 3.3, the distribution is not scaled)

$$(y[1], y[2]) \longmapsto \exp\left(-\frac{y[1]^2}{200} - \frac{1}{2}[y[2] - 0.03(y[1]^2 - 100)]^2 \right)$$

To sample from this distribution, I implemented an Hamiltonian Monte Carlo algorithm which I used as groundtruth.

Should the likelihood be an independant product of normal distribution of standard deviation 1, some observations would have a very low likelihood and the estimation would fail. That's why I multiplied all the length in this section.

I adapted the move step and the proposition distribution. The proposition distribution is now an independant product of two normal distributions, with means the empirical means and with variances the empirical variances.

I implemented IBIS for the two dimensional observations data and the two dimensional parameters (particles).

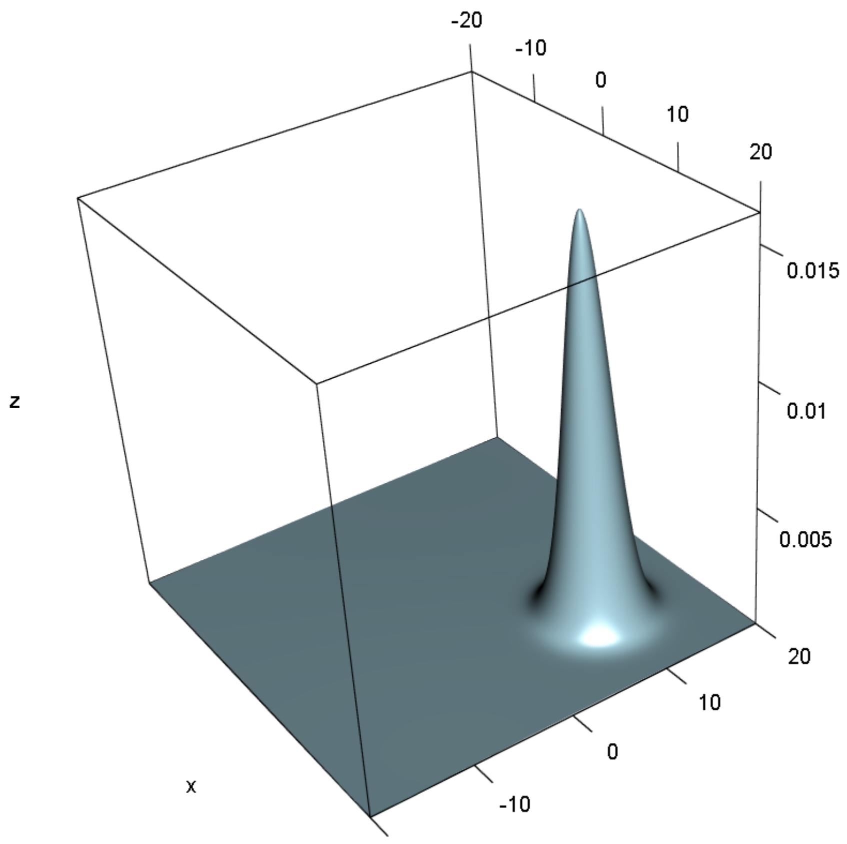

Figure 3.2: Mixture model in dimension two

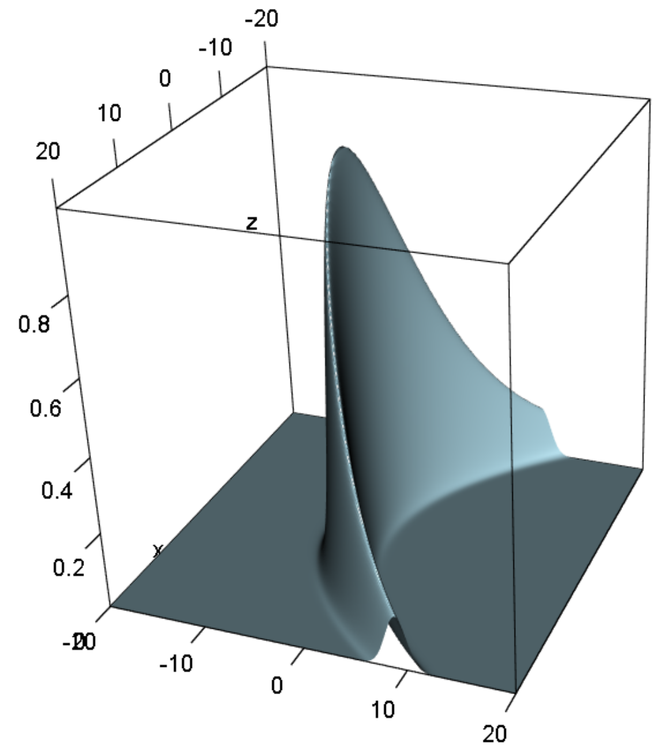

Figure 3.3: Banana in dimension two

3.4.2 Principal Component Analysis (PCA)

In dimension one, the best method is to choose the batchs by quantiles. If we want to use this idea in dimension $d > 1$ we have to use an order on $\mathbf{R}^d$. Rather than using the lexical order on $\mathbf{R}^d$, I reduced the dimension of the data with a Principal Component Analysis (PCA) and projected the data on the more representative axis of this PCA. Then, I applied the method I used on the real line on this axis.

We are in a simple case of PCA. We have a cloud of points $x_1, ..., x_N$, which is supposed to be centered $x_1 + ... + x_N = 0$. Let M be the matrix of the cloud, $i.e$ the column i of M is the coordinates of the x_i in the canonical base of $\mathbf{R}^d$. We want to find "the" subspace of dimension 1 the closer to the cloud of points, that is we want to solve

$$\inf_D \sum_{j=1}^N \|x_j - p_D(x_j)\|^2$$

where the inf is taken on all subspaces D of dimension 1 and, p_D is the orthogonal projector on D. For all j and all subspaces D we have

$$\|x_j - p_D(x_j)\|^2 + \|p_D(x_j)\|^2 = \|x_j\|^2$$

and the quantity $\sum_{j=1}^N \|x_j\|^2$ is fixed in our problem so we just have to solve

$$\sup_D \sum_{j=1}^N \|p_D(x_j)\|^2.$$

Let D be a subspace of dimension 1 and let a be a normalized vector in D. Then,

$$\sum_{j=1}^N \|p_D(x_j)\|^2 = \sum_{j=1}^N (a|x_j)^2 = \sum_{j=1}^N (a|x_j)_d^2 = (M^t a|M^t a)_N = a^t M M^t a$$

where $(.|.)_k$ is the scalar product on $\mathbf{R}^k$ and M^t is the transpose matrix of M. Finally we have to solve

$$\sup_{\|a\|=1} a^t M M^t a.$$

MM^t is a real positive symmetric matrix so we know that an eigenvector, say e, for the highest eigenvalue, say λ, satisfies

$$\sup_{\|a\|=1} a^t M M^t a = \lambda = e^t M M^t e.$$

Then, if $D = Span(e)$ we compute $p_D(x_1), ..., p_D(x_N)$ and we can rank the data by quantiles replacing the real line by D. The method is expected to be fast because there are powerful algorithms to perform the PCA but by projecting the data on one axis we loose a lot of information about it, even if the axis is the better axis.

By drawing without replacement		By performing a PCA	
Mean	Variance	Mean	Variance
0.0176	0.1882	0.0563	0.1186
0.0076	0.2255	0.0052	0.1217
0.1293	14.5209	0.5874	10.4158
0.0259	14.7266	0.0901	9.6481
0.0523	6.3279	0.0903	3.292011

Table 3.2: Results in dimension two for the mixture model

By drawing without replacement		By performing a PCA	
Mean	Variance	Mean	Variance
0.1072	0.1498	0.0255	0.2888
0.1165	0.3337	0.1269	0.1698
0.3574	2.2984	0.2337	3.8184
2.0019	191.8019	2.8221	82.7178
1.3501	26.27129	0.4125	37.2220

Table 3.3: Results in dimension two for the banana

3.4.3 Results

I compared the results of the two methods by computing

$$|\frac{1}{n}\sum_{i=1}^{n}\theta[j]_i - \int_{\Theta}\theta[j]\,\pi(\mathrm{d}\theta|Y_1,Y_2,\ldots,Y_N)|$$

at each simulation for $j = 1$ (see the first line of the tables 3.2 and 3.3) and for $j = 2$ (see the second line of the tables 3.2 and 3.3). I computed

$$|\frac{1}{n}\sum_{i=1}^{n}\theta[j]_i^2 - \int_{\Theta}\theta[j]^2\,\pi(\mathrm{d}\theta|Y_1,Y_2,\ldots,Y_N)|$$

at each simulation for $j = 1$ (see the third line of the tables 3.2 and 3.3) and for $j = 2$ (see the fourth line of the tables 3.2 and 3.3). I also computed the difference of the empirical and the theoretical crossed moments (see the fifth line of the tables 3.2 and 3.3)

$$|\frac{1}{n}\sum_{i=1}^{n}\theta[1]_i\theta[2]_i - \int_{\Theta}\theta[1]\theta[2]\,\pi(\mathrm{d}\theta|Y_1,Y_2,\ldots,Y_N)|.$$

We expected the PCA method to be better than drawing without replacement. It is not clear with those results. For the sampling from a banana the means for the PCA method are sometimes smaller than the means for the drawing without replacement, for example in the first and the last lines. The means are more than three times smaller but the variances are higher. For the sampling from a mixture of normal distribution we cannot say that the PCA methods is globally better than the drawing without replacement with these results.

Chapter 4

Stochastic Approximation online Expectation-Maximization (SAoEM) for Hidden Markov models (HMM)

4.1 Expectation-Maximization (EM) algorithm

Assume we observe two random variables X and Y. X is the missing data and Y the observed data. Assume that (X, Y) is drawn from P_θ where $\theta \in \Theta$ and Θ is a set of deterministic finite-dimensional parameter. The likelihood in this model is

$$f(X, Y, \theta)$$

and the available likelihood is

$$L(Y, \theta) = \int f(x, Y, \theta)\,\mathrm{d}x.$$

Our aim is to perform a likelihood maximization for θ. We define the conditional probability density function of X given Y

$$p(x|Y, \theta) = \frac{f(x, Y, \theta)}{L(Y, \theta)},$$

the available log-likelihood

$$l(Y, \theta) = \log(L(Y, \theta))$$

and the intermediate quantity of the EM algorithm

$$Q(Y, \theta, \theta') = \int \log(f(x, Y, \theta))p(x|Y, \theta')\,\mathrm{d}x = \mathbf{E}_{\theta'}(f(X, Y, \theta)|Y)$$

The intermediate quantity will be useful to perform the likelihood maximization because of the fundamental inequality :

$$l(y,\theta) - l(y,\theta') \geq Q(y,\theta,\theta') - Q(y,\theta',\theta')$$

which is an equality if and only if

$$\theta = \theta'.$$

The Expectation-Maximization algorithm builds a sequence (θ_i) of parameter estimates given an initial guess θ_0. The iteration i of the algorithm is divided in two steps as follows :

- Expectation step (E-step) : Compute $\theta \longmapsto Q(y,\theta,\theta_i)$
- Maximization step (M-step) : Choose θ_{i+1} among the values that maximize the function in the E-step.

The idea in this algorithm is that each iteration will determine a value θ_{i+1} which increases the log-likelihood because of the fundamental inequality.

4.2 Online EM

Assume that $X = (X_i)_i$ and $Y = (Y_i)_i$ where $(X_1, Y_1), ..., (X_N, Y_N)$ are independant and identically distributed from P_θ (note that in the previous section P_θ was the distribution of (X, Y), but if P_θ is fully determined by θ then the distribution of (X, Y) is fully determined by θ since the observations are i.i.d).

$$Q_n(Y,\theta,\theta') = \frac{1}{n}\mathbf{E}_{\theta'}(\log(L_n(X,Y,\theta))|Y) = \frac{1}{n}\sum_{i=1}^{n}\mathbf{E}_{\theta'}(\log(L(X_i,Y_i,\theta))|Y_i)$$

where L_n is the likelihood for n observations and Q_n the intermediate quantity for n observations normalized by n. We can link the intermediate quantity for n observations to the intermediate quantity for $n+1$ observations.

$$Q_{n+1}(Y,\theta,\theta') = Q_n(Y,\theta,\theta') + \frac{1}{n+1}(\mathbf{E}_{\theta'}(\log(L(X_{n+1},Y_{n+1},\theta))|Y_{n+1}) - Q_n(Y,\theta,\theta'))$$

In ([4]) the author uses the formula to derive an online EM algorithm where the E-step is online and is about computing an approximation of the intermediate quantity. The M-step is the maximization of the approximate intermediate quantity. The iteration n is

- E-step : Compute
$$\widehat{Q}_{n+1} : \theta \longmapsto \widehat{Q}_n(\theta) + \frac{1}{n+1}(\mathbf{E}_{\widehat{\theta}_n}(\log(L(X_{n+1},Y_{n+1},\theta))|Y_{n+1}) - \widehat{Q}_n(\theta))$$
- M-step : Choose $\widehat{\theta}_{n+1}$ among the values that maximize the function in the E-step.

In general, the expectation $\mathbf{E}_{\widehat{\theta}_n}(\log(L(X_{n+1},Y_{n+1},\theta))|Y_{n+1})$ cannot be computed in the closed form. Therefore in [6] the authors add an simulation step before the E-step. In this step they sample a reversible Markov chain having $P_{\widehat{\theta}_n}|Y_{n+1}$ as its unique stationnary distribution and they approximate this expectation by an ergodic mean. This approach was introduced in [2] in a static framework (not online) and the SAEM algorithm still converges.

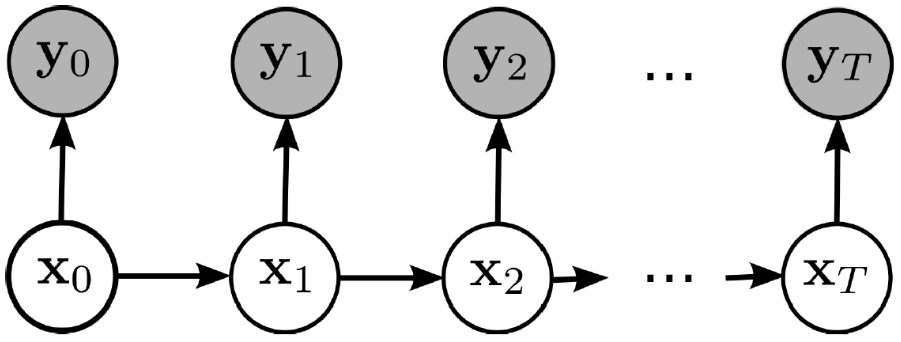

Figure 4.1: Hidden Markov model

4.3 Hidden Markov models (HMM)

"A hidden Markov model (HMM) is, loosely speaking, a Markov chain observed in noise." (see [3] for an introduction to HMM.) It's a stochastic process $(X_k, Y_k)_{k\geq 0}$ where (X_k) is a Markov chain (which is hidden) and for all $k \geq 0$, Y_k is an observed random variable which only depends on X_k. In several cases a HMM $(X_k, Y_k)_{k\geq 0}$ can be represented like that :

$$\begin{cases} X_{k+1} = a(X_k, U_k) \\ Y_k = b(X_k, V_k) \end{cases}$$

where (U_k) and (V_k) are sequences of independant variables, independant on X_0, and a and b are measurables functions. The first equation gives the dynamic of the model and the second gives the observation scheme.

It can be represented as follows (4.1).

A particular case of HMM, a fully dominated HMM, is a Markov chain $(X_k, Y_k)_{k\geq 0}$ (taking its values in a product of measurable spaces $\chi \times \mathbf{R}$) which kernel has density according to $\mu \times \lambda$, t and can be written

$$t((x,y),(x',y')) = q(x,x')g(x',y)$$

where q is the density of a transition kernel of the Markov chain (X_k) from χ to χ, g the density of a conditional transition kernel from χ to $\mathbf{R}$, μ a
erence measure on χ and λ the Lebesgue measure.

4.4 SAoEM for HMM

We observe a (part of a) HMM $(X_1, Y_1), ..., (X_N, Y_N)$ where $X = (X_i)_i$ is the hidden data and $Y = (Y_i)_i$ the observed data. Assume that the HMM is entirely determinated by a finite-dimensional parameter θ. We have

$$L_n(X,Y,\theta) = \nu_\theta(X_1)g_\theta(X_1,Y_1)q_\theta(X_1,X_2)...q_\theta(X_{n-1},X_n)g_\theta(X_n,Y_n)$$

with the same notations, where ν_θ is the distribution of X_1 and is entierly determinated by θ.

Then,

$$nQ_n(Y,\theta,\theta') = \mathbf{E}_{\theta'}(\log \nu_\theta(X_1)|Y) + \sum_{i=2}^{n} \mathbf{E}_{\theta'}(\log p_\theta(X_i,Y_i|X_{i-1})|Y)$$

where $p_\theta(x,y|x') = q_\theta(x',x)g_\theta(x,y)$. In the i.i.d case we could say that the Y_i are i.i.d and simplify the conditonal expactation to derive an equality between Q_{n+1} and Q_n. Here, we cannot generalize directly this equality because we observe a HMM. One can check that the following formula for the particular case of HMM where the observations are i.i.d is exactly this equality. We forget the dependance on the Y_i in the notations and write $Y_{0:n}$ for the sequence $Y_1,...,Y_n$.

$$\phi_{n,\theta'}(x) = P_{\theta'}(X_n = x|Y_{0:n})$$

and

$$\rho_{n,\theta'}(x,\theta) = \frac{1}{n}\mathbf{E}_{\theta'}(\sum_{i=1}^{n} \log p_\theta(X_i,Y_i|X_{i-1})|Y_{0:n}, X_n = x)$$

satisfy

$$\sum_{x\in\chi} \phi_{n,\theta'}(x)\rho_{n,\theta'}(x,\theta) = \frac{1}{n}\mathbf{E}_{\theta'}(\sum_{i=1}^{n} \log p_\theta(X_i,Y_i|X_{i-1})|Y_{0:n}).$$

We also have

$$\phi_{0,\theta'}(x) = \frac{\nu(x)g_{\theta'}(x,Y_0)}{\sum_{x'\in\chi}\nu(x')g_{\theta'}(x',Y_0)}, \rho_{0,\theta'}(x) = 0$$

and

$$\forall n \geq 1, \phi_{n+1,\theta'}(x) = \frac{\sum_{x'\in\chi}\phi_{n,\theta'}(x')q_{\theta'}(x',x)g_{\theta'}(x,Y_{n+1})}{\sum_{x',x''\in\chi}\phi_{n,\theta'}(x')q_{\theta'}(x',x'')g_{\theta'}(x'',Y_{n+1})}$$

and

$$\rho_{n+1,\theta'}(x) = \sum_{x'\in\chi}\left(\frac{1}{n+1}\log p_\theta(x,Y_{n+1}|x') + (1-\frac{1}{n+1})\rho_{n,\theta'}(x',\theta)\right) P_\theta(X_n = x'|X_{n+1} = x, Y_{0:n})$$

where

$$P_\theta(X_n = x'|X_{n+1} = x, Y_{0:n}) = \frac{\phi_{n,\theta'}(x')q_{\theta'}(x',x)}{\sum_{x''\in\chi}\phi_{n,\theta'}(x'')q_{\theta'}(x'',x)}$$

In [5] the author derives from this formula an online EM for HMM in the same way he did for the i.i.d case. If the hidden data of the HMM takes its values in a general measurable space χ (not denombrable) and if the HMM is fully dominated, we can rewrite this formula by replacing the sums by integrals and the probabilities by densities. Then we can derive an online EM for HMM in a general measurable space. The E-step will be online:

$$\widehat{\phi}_0(x) = \frac{\nu(x)g_{\widehat{\theta}_0}(x,Y_0)}{\int_\chi \nu(\mathrm{d}x')g_{\widehat{\theta}_0}(x',Y_0)}, \widehat{\rho}_0(x,\theta) = 0$$

and

$$\forall n \geq 1, \widehat{\phi}_{n+1}(x) = \frac{\int_\chi \widehat{\phi}_n(\mathrm{d}x') q_{\widehat{\theta}_n}(x', x) g_{\widehat{\theta}_n}(x, Y_{n+1})}{\int_\chi \int_\chi \widehat{\phi}_n(\mathrm{d}x') q_{\widehat{\theta}_n}(x', \mathrm{d}x'') g_{\widehat{\theta}_n}(x'', Y_{n+1})}$$

$$\widehat{\rho}_{n+1}(x) = \int_\chi \left(\frac{1}{n+1} \log p_\theta(x, Y_{n+1}|x') + (1 - \frac{1}{n+1}) \widehat{\rho}_n(x', \theta) \right) \widehat{\pi}_n(x, \mathrm{d}x')$$

where

$$\widehat{\pi}_n(x, x') = \frac{\widehat{\phi}_n(x') q_{\widehat{\theta}_n}(x', x)}{\int_\chi \widehat{\phi}_n(\mathrm{d}x'') q_{\widehat{\theta}_n}(x'', x)}$$

and finally the M-step will be the maximization of

$$\theta \longmapsto \int_\chi \widehat{\phi}_n(\mathrm{d}x) \widehat{\rho}_n(x, \theta)$$

If we try to use the approach of [6], we will try to approximate the expectactions $\int_\chi (\log p_\theta(x, Y_{n+1}|x'))\, \widehat{\pi}_n(x, \mathrm{d}x')$ by ergodic means, **for all** $x \in \chi$, whereas in the i.i.d case there is also one expectation to approximate. Therefore it is impossible to generalize directly the approach used by the authors of [6].

Chapter 5

Conclusion

The IBIS algorithm processes independant data of a distribution parametrized by a random vector. Its aim is to sample from the posterior distribution of this random vector. I tried different ways to rank the data in a pre-processing step and measured the sampling efficiency for each method. For real observations, the best method is to rank the data by quantiles. I also try to rank the data by score of likelihood. As expected it deteriorates the quality of the sampling. In dimension one, the ranking of data, when it's possible, has a striking impact on the result of the algorithm. For multi-dimensional data, particularly for two dimensional data, I compared two methods of ranking. The first is the drawing without replacement, that is to process the observations in the order they appear. The second is to reduce the dimension of the observations by performing a principal component analysis and use the best method in dimension one, the ranking by quantiles, on the best axis. I expected this one to be better than the other but it is not clear on my simulations.

Then I worked on the generalization of an online EM algorithm for observations following a Hidden Markov model. I tried to generalize it to the case where the hidden data in the HMM takes its values in a general measurable space. Conditional expectations have to be computed. But for all points of this general measured space we have one conditional expectation to compute so we cannot approximate them by an ergodic mean in general.

We can expect the likelihood method of ranking to be worse in dimension $d > 1$ than in dimension one. An other bad method of ranking in dimension one could be to rank the data by increasing order. These methods show that the ranking of the data does have an impact on the results. In high dimension we can use a clustering method and make the batchs by drawing one element in each cluster, in an attempt to mimic the quantiles ranking method in dimension 1. Clustering methods could be useful for mixture models. However with such a method we do not control the number of observations by batch because the number of observations by cluster is not the same in all clusters. Moreover, a cluster could be empty at the end of the algorithm (for example in the k-means method). Compared to the PCA method, a clustering could have better results but clustering would be more computationally demanding.

The next step of this work is to try to perform the PCA method in very high dimension and to compare it to the drawing without replacement. Maybe one should change the move step for high dimensional to avoid the estimation of the covariance matrix of the particles.

Bibliography

[1] N. Chopin. (2002), "A sequential particle filter for static models" *Biometrika*, 89:539-552

[2] L. M. Delyon, B. and E. Moulines (1999), "Convergence of a stochastic approximation version of the EM algorithm," *The Annals of Statistics*, vol. 27, pp. 94–128

[3] Cappé, O., Moulines, E., and Rydèn, T. (2005), Inference in Hidden Markov Models, *Springer*

[4] Cappé, O. and Moulines, E. (2009), "On-line expectation-maximization algorithm for latent data models," *J. Roy. Statist. Soc. B*, 71(3):593–613

[5] Cappé, O. (2011), "Online EM Algorithm for Hidden Markov Models," *J. Comput. Graph. Statist*, 20(3):728–749

[6] Maire, F.; Moulines, E.; Lefebvre, S. (2013), "Online EM for Functional Data ," *IEEE Transactions on Pattern Analysis and Machine Intel*

[7] Glaeser, G. (1963), "Fonctions composéès differentiables ," *Ann. Math.*, 77, 193–209

[8] Girolami, M., Calderhead, B. (2011), "Riemann manifold Langevin and Hamiltonian Monte Carlo methods" *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, , 73(2): 123–214

[9] Neal, R. M. (2010), "MCMC using Hamiltonian dynamics" *Handbook of Markov Chain Monte Carlo*

[10] Del Moral, P., Doucet, A., and Jasra, A. (2006), "Sequential Monte Carlo samplers" *J. R. Statist. Soc. B*, 68(3):411–436.