Metropolis-Hastings Algorithm

Justine Weber, Lucie Perrotta, Robin Leurent

December 2018

Abstract

The theory related to Markov Chains is used in practice in many fields, in particular for algorithms, and in statistical physics. In this report, we will show how the Metropolis-Hasting algorithm, a Markov Chain Monte-Carlo (MCMC) method, can be used to solve a non-linear estimation problem, and how it can be optimized to perform better and achieve faster convergence. In particular, we will explain how we implemented simulated annealing and develop an efficient cooling strategy, taking in account the limited available computational power.

1 Introduction

1.1 Disclaimer

In order to make the computations feasible, we have bounded the variables to some values ranges. Our results have been computed with 5000 iterations, while n has been set to 1000, as suggested in the course. m takes values between 0 and 10'000 so that $\alpha = \frac{m}{n}$ lies between 0 and 10.

1.2 Non-linear estimation problem

Non-linear estimation does compute the relationship between a known and fixed set of variables, and a - to be found - set of unknown variables. The MCMC method is used to solve this kind of problem in a linear way (1). We typically consider an unknown vector $\mathbf{X} = (X_1, ..., X_n)^T \in S = \{-1, 1\}^n$, where $n \sim 1000$. We will estimate $\mathbf{X}$ using a matrix $W_{ij} \overset{iid}{\sim} \mathcal{N}(0, 1)$, where the order of magnitude of n typically lies between $\frac{m}{10}$ and m.

We observe the vector $\mathbf{Y} = \text{ReLU}\left(\left[\frac{W\mathbf{X}}{\sqrt{n}}\right]\right)$, where ReLU is the *Rectifier Linear Unit* function, applied element-wise to the vector. The aim of this problem is to minimize the energy, defined as the squared Euclidean distance between $\mathbf{Y}$ and $\mathbf{Y'}$, a vector computed like $\mathbf{Y}$ but with a random vector $\mathbf{x} \in S = \{-1, 1\}^n$ instead of the ground truth $\mathbf{X}$. We hence use MCMC to change values in $\mathbf{x}$ to turn it into $\mathbf{X}$ so to minimize the energy.

2 Model overview

2.1 The metropolis algorithm

The model is based on the Metropolis Algorithm. As explained above, given an vector $\mathbf{X}$, and matrix W, we start with a random initial vector x_0, and aim to make it evolve to converge the closest to $\mathbf{X}$. The process of making x_0 evolve works as follows. At each iteration t, a random index i is drawn. Let x'_t be such that:

$$\begin{cases} x'_t[i] = -x_t[i] \\ x'_t[j] = x_t[j], \text{ for all } j \neq i \end{cases}$$

We define the acceptance probability:

$$a = min(1, e^{-\beta(H_Y(x_1) - H_Y(x_0))}) \tag{1}$$

where H is the energy between the truth observation and the x observation. At each iteration, x_{t+1} takes value x'_t with probability a. Otherwise x_{t+1} takes value x_t, i.e. we change the sign of one element of vector x_{t+1} form vector x_t, with probability a.

At each iteration, we also compute the error between x_t and the truth $\mathbf{X}$. We iterate for as long as the error is strictly positive or if we reach the maximum number of iterations, chosen arbitrarily to be high enough. We will discuss the meaning of the error in a later section.

2.2 Parameters estimation

We have experimented on several hyper-parameters in order to have an optimal algorithm. First, we chose a small initial β_0 of 0.2 in order to have a high acceptance probability on the first iterations. Then, as proposed in (2), we increase β linearly by a step of 0.2 if two conditions are satisfied simultaneously:

- First, the variance of the last 200 iterations needs to be inferior to $\frac{m}{285.7}$. The value of 285.7

has been found experimentally as we noticed that for $m = 2000$, the optimal standard deviation for updating α was around 7, which equals $\frac{1000}{285.7}$. Generalizing over m, we set it to $\frac{m}{285.7}$. This ensures that we begin to increase β only if we reach a stagnation phase, i.e. if there is no significant possible improvement left, by keeping the current β.

- Secondly, the number of iterations between two successive increases of β must be at least $\frac{max\_iter}{40}$. This allows us to speed up the convergence to a reasonably small error even with only 5000 iterations.

3 Optimization Process

3.1 Cooling strategy

The double condition for the update of β leads to two successive phases is the evolution of the temperature.

The first phase corresponds to the iterations when satisfying the first condition above takes longer than the second one. We hence update the beta only when the slope of the energy has become smooth and flat enough. Since the threshold is computed as a function of m and of the energy, we denote this phase the **smart phase**. Looking at figure 1, one can clearly notice the jump in the energy around iteration 1700, which corresponds to a change of the beta, right after the energy curve had begun to slow its descent.

Note that the standard deviation of the energy decreases during the period corresponding to one value of β, but it also decreases globally during the full convergence of the algorithm. This means that after changing from one beta to the next one, the standard deviation of the energy will start at a smaller value than for the previous beta. Consequently, the smart phase update of the beta will cause the beta to be updated more and more often after some iterations, as the standard deviation is constantly below the threshold after this time. That's where the second condition on the update becomes important. It imposes a maximum update speed, so to keep the update slow enough, and leave some time for the energy to decrease before changing β again. We call this quick update phase the **stairs phase**.

The stairs phase allows the temperature not to cool down too quickly and to fall into a local minimum. The figure 1 shows a typical behaviour of the beta.

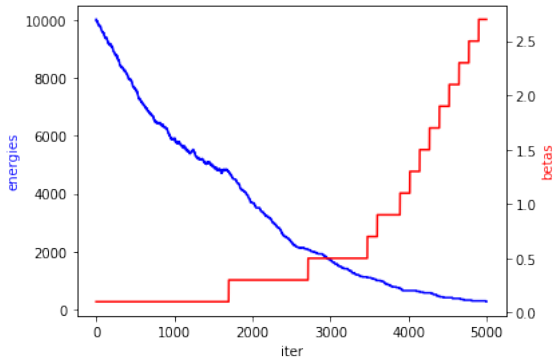


Figure 1: Evolution of the beta for $\alpha = 10$. One can clearly see the transition from smart to stairs phase around iteration 3800.

3.2 Reconstruction error

Minimizing the energy amounts to minimizing the error. Here, we give a sensitive meaning to the search for a minimum error.

Claim *The error $e(\hat{x}, \boldsymbol{X})$ is the fraction of entries on which $\hat{x}$ differs from $\boldsymbol{X}$.*

Proof. Let's define z as the number of positions in which $\hat{x}$ and $\mathbf{X}$ differ, and let's remind that the length of both $\hat{x}$ and $\mathbf{X}$ is n.

$$\begin{aligned} e(\hat{x}, \mathbf{X}) &\triangleq \frac{1}{4n}\|\hat{x} - X\|_2^2 \\ &= \frac{1}{4n}\sum_{i=1}^{n}(\hat{x}_i - X_i)^2 \\ &= \frac{1}{4n}\left(\sum_{i:\hat{x}_i = X_i} 0^2 + \sum_{i:\hat{x}_i \neq X_i}(\pm 2)^2\right) \\ &= \frac{1}{4n}\sum_{i:\hat{x}_i \neq X_i} 4 = \frac{1}{4n}4z = \frac{z}{n}. \end{aligned}$$

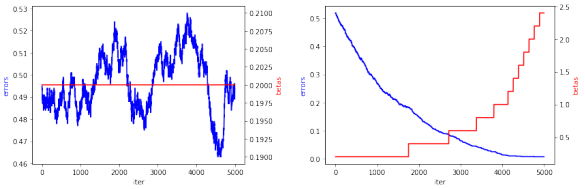
Clearly, $\frac{z}{n}$ is the fraction of entries on which $\hat{x}$ differs from $\mathbf{X}$. □

We hence see that minimizing the error corresponds to converging to the objective vector $\mathbf{X}$.

4 Results

We see with no surprise that the larger the α, the faster the error converges to 0. Actually, when α is small ($\alpha < \frac{1}{2}$), the error stays very high and does not converge even for a large number of iterations, as shown in figure 2. In this extreme case, the standard deviation of the energy is constantly so high that the beta does not even change once before we reach the maximum number of iterations and stop the experiment.

Conversely, when α is large ($\alpha > e$), convergence of the error to 0 is almost certain, and depends on the number of iterations, as the cooling strategy does.



(a) $\alpha = 0.01$ (b) $\alpha = 10$

Figure 2: Comparison of the evolution of the error and of β for different values of α, and 5000 iterations. One can clearly see that convergence is not achieved for small alpha, as it is for large alpha.

We can hence plot the mean value and the standard deviation of the error for increasing values of α. this is shown in figure 3. As suggested from figure 2, the mean value of the error decreases as α increases. However, the decreasing curve of the mean error is not regular: it decreases pretty fast from $\alpha = 0$ to around $\alpha = e$, where it reaches an error of approximately 0, and then stays around 0 for greater values of α. This values corresponds hence to α_c, where the performance of the algorithm changes abruptly from improving with α up to the perfect converge, to staying at its best.

Respectively, the standard deviation of the error takes various values for $\alpha < e$, and even increases as α increases, to then suddenly drop when $\alpha \sim e$ and keep lower and more regular values for larger α.

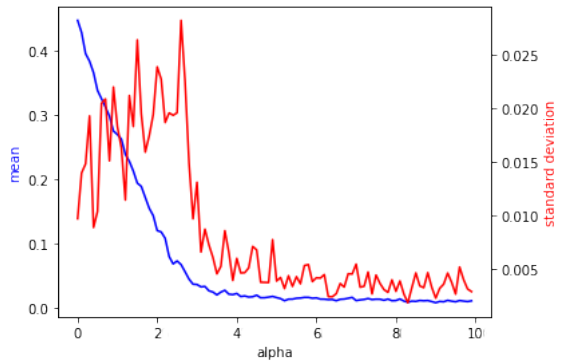


Figure 3: Evolution of the mean and the standard deviation for $0 < \alpha \leq 10$ and 5000 iterations. Notice the big jump in the standard deviation at $\alpha \sim e$.

We can also see that the mean of the error tends to 0.5 as α tends to 0. When decreasing the number of iterations, one gives less time to the error to converge and may not observe a convergence within the iterations. For instance, with 5000 iterations, the minimum value for a convergence to happen is around $\alpha_{rdm} \sim 0.66$, but when increasing the number of iterations, the number α_{rdm} decreases until it reaches negligible values for very a large number of iterations.

5 Conclusion

We have seen how we can use the Metropolis algorithm coupled to a simulated annealing strategy in order to predict a vector X with an error rate of almost almost 0% as long as α is large enough. We could have predicted faster this vector with more powerful computers or a higher α. The higher alpha was, the longer the computations took. With limited computer power, we hence dealt with a trade-off between the size of α and the number of iterations. Convergence to low error for small α is possible but was never observed in our experiments as the number of iterations does not let enough time to the vector $\hat{x}$ to converge to the truth.

References

[1] O. Leveque and N. Macris, *Markov Chains and Algorithmic Applications Project 2018.* `https://moodle.epfl.ch/pluginfile.php/2459473/mod_resource/content/3/project_MCAA_1819.pdf`

[2] Y. Nourani and B. Andresen, *A comparison of simulated annealing cooling strategies.* Formula (3). `https://www.fys.ku.dk/~andresen/BAhome/ownpapers/permanents/annealSched.pdf`