Tutorial 7

Maciej Wozniak

March 4, 2022

1 Introduction

The tutorial introduced different techniques for sampling and estimating difficult probability distributions, by approximating them using simpler ones. In the practical part we focused on Metropolis-Hastings and Bootstrap Filter. Solutions and short descriptions of each algorithms can be found below.

2 Metropolis-Hastings exercise

Metropolis - Hasting algorithm allows us on sampling from complex distribution but approximating it with simple one, such as Normal distribution. It builds a Markov chain where the transition from one node to another is connected with the new sample being accepted.

Simple sudo code:

1. Initial values for x_0 (params)
2. for i in range(m):
   (a) draw a candidate x_i (from a proposal distribution) $x_i\ q(x_i|x_{i-1})$
   (b) Calculate acceptance rate $\alpha = \frac{p(x_i)*q(x_{i-1}|x_i)}{q(x_i|x_{i-1})*p(x_{i-1})}$
      - if $(\alpha >= 1)$
        accept x_i and $x_{i-1} \leftarrow x_i$
      - elif $(0 < alpha <= 1)$
        reject x_i

Experiments were set it up in such as way that:

- We were sampling from Gausian Mixture model described by

$$p(x)\ \alpha\ a_1\mathcal{N}(x|\mu_1, \sigma_1^2) + a_2\mathcal{N}(x|\mu_2, \sigma_2^2) \tag{1}$$

- parameters were set as $a_2 = a_2 = 0.5,\ \mu_1 = 0,\ \mu_2 = 3,\ \sigma_1 = 1,\ \sigma_2 = 0.5$
- We used different variance for sampling 0.1,1, 10, and 100 testing their accuracy of predicting the target distribution.

In the figures below we can see how different *scale* (variances) influence the accuracy of a predicted probability distribution. It can be observed that using the smaller variance, creates distribution closely correlated with the GT where all the samples are near placed very close to the mean values. With higher variance (100) estimated pdf is much smoother than the one predicted with the variance of 0.1. We could also see that with smaller variance, we often reject the samples for many *iterations* 1, whereas with the higher variance samples are accepted more frequently. Another observation we could make is how far they are from the mean (which is basically what covariance of the distribution tells us). We can see that when we sample with smaller covariance we tend to have much more samples close to the mean values and the higher covariance we have, the more equally distributed the samples are. Choosing the perfect variance for sampling depends on the distribution we are sampling from and choosing the best one would be a separate optimization task itself. Then the higher the variance is, the more I used 10 000 samples and generated *numpy* array of these samples, drawing from the pdf's generated by *scipy*. The code was attached to the submission and comments are added to facilitate understanding of the workflow.

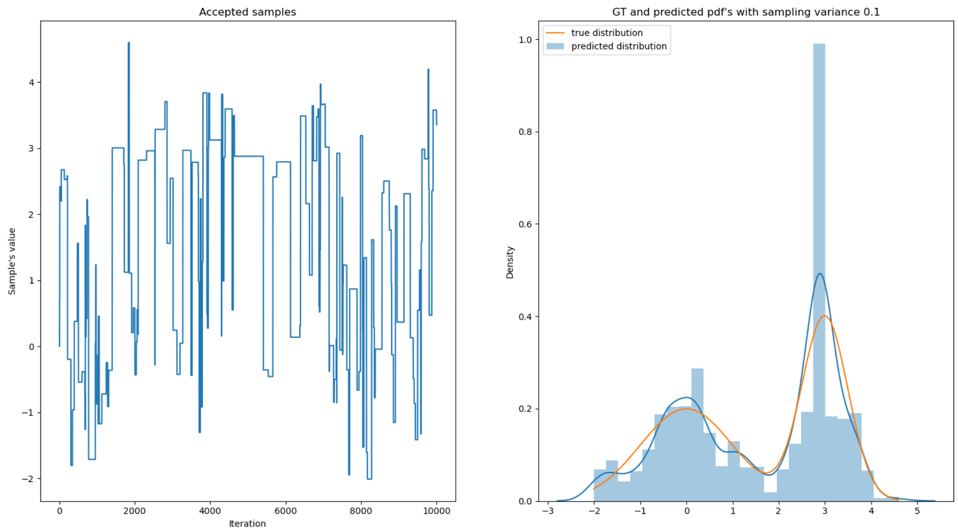


Figure 1: Variance = 0.1

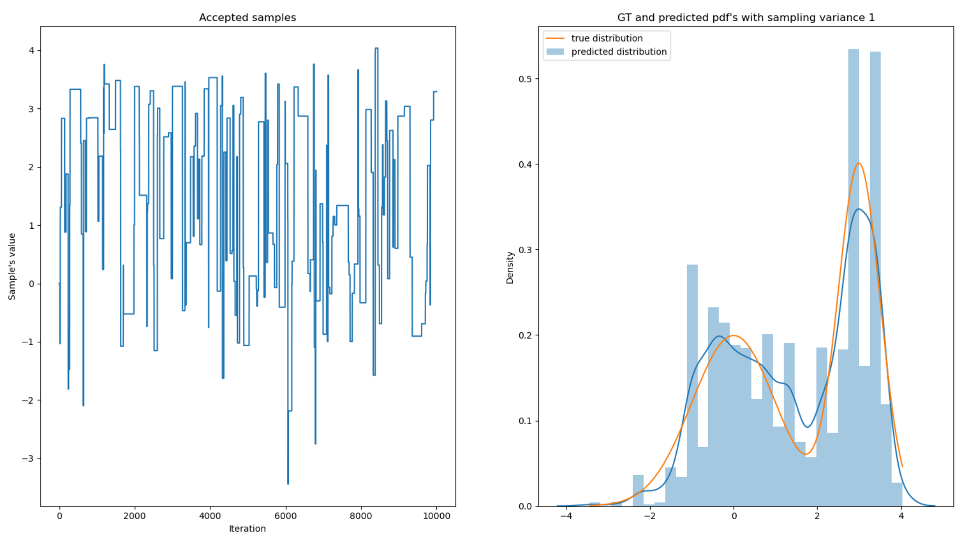


Figure 2: Variance = 1

3 Bootstrap filter exercise

Bootstrap filter is used to estimate complex distributions, correcting them each iteration. It starts with knowledge about the prior distribution $p(x)$. We initiate the algorithm by sampling N samples, equal to the number of particles from that distribution. At each step, we will extract N samples from x_t $p(x_k|x_{k-1})$ and calculate their weights. Next, we will choose N samples from the *sample* with the

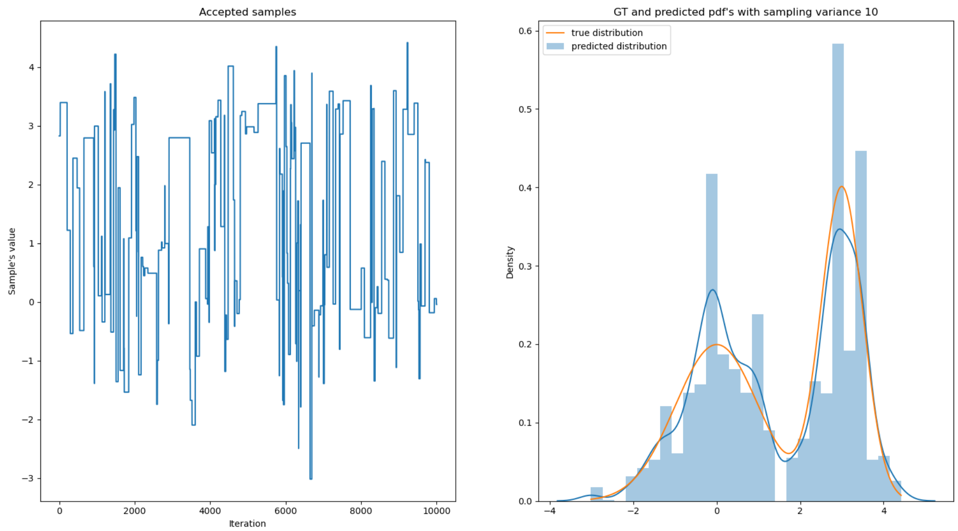


Figure 3: Variance = 10

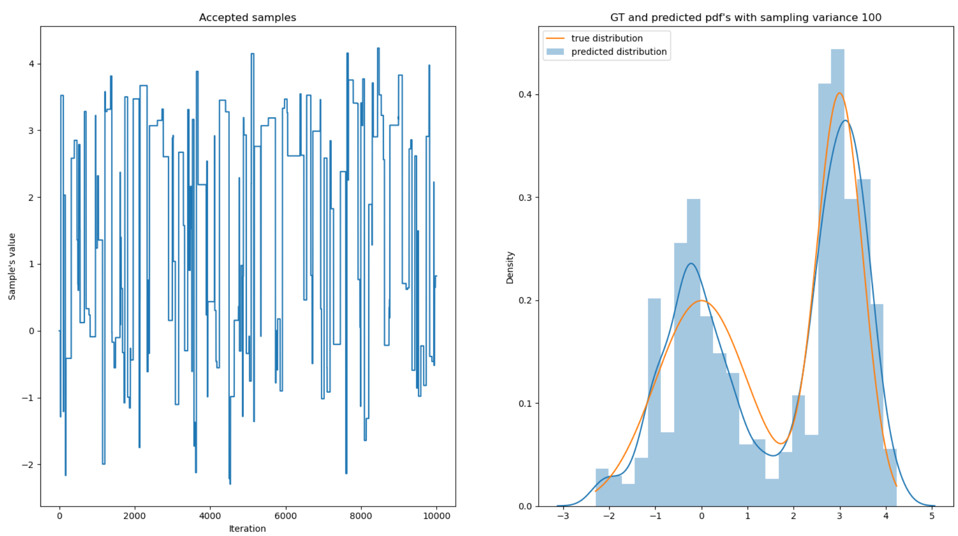
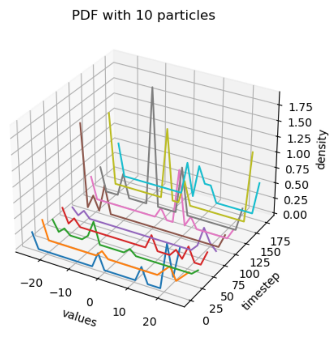


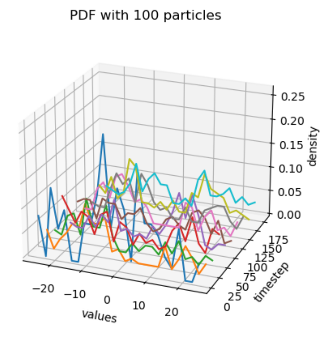
Figure 4: Variance = 100

probability according to the weights. Note that the same sample could be chosen multiple times and this feature causes the *PDF* to *move around*. As showed in Figure 5 below, I started with tested different number of particles - 10, 100, 1000, 10000, 100000, 1000000. Some multimodality can be already seen in the plot where I used 100 particles, but to notice it clearly we would need at least 1000 particles. It also depends on how many *bins* we are using while creating the plots (I used 25). In the beginning increasing number of particles positively influences the estimations but after *10000* particles, we could not see a significant improvement anymore. In short:

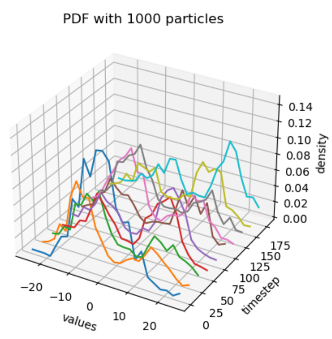
- Question 1 100/1000
- Question 2 N = 10000



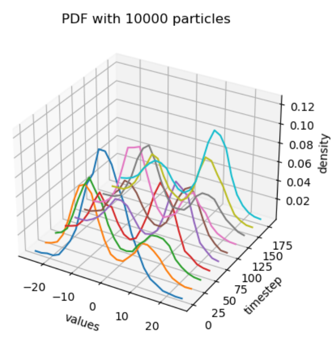
(a) 10 Particles



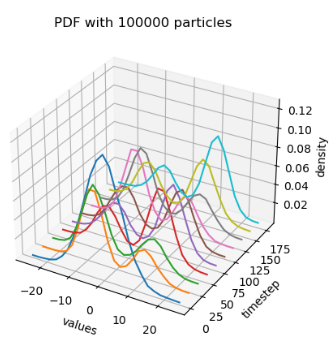
(b) 100 Particles



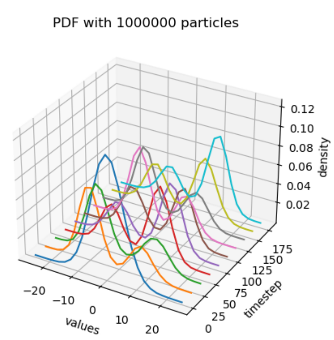
(c) 1000 Particles



(d) 10000 Particles



(e) 100000 Particles



(f) 1000000 Particles

Figure 5: Boostrap filter with n particles.