Learning Summary Statistics for Approximate Bayesian Computation

Yiwen Chen,
Email: yiwen15@stanford.edu

***Abstract*—In high dimensional data, it is often very difficult to analytically evaluate the likelihood function, and thus hard to get a Bayesian posterior estimation. Approximate Bayesian Computation is an important algorithm in this application. However, to apply the algorithm, we need to compress the data into low dimensional summary statistics, which is typically hard to get in an analytical form. In this project, I used Radial Basis Function network and Neural Network to learn the summary statistics automatically. I constructed a time series example and a hidden markov chain example, and I find that Neural network performs better.**

I. Introduction

In Bayesian Estimation, one of the most important step is to calculate the likelihood function $p(X|\tilde{\theta} = \theta)$, where $\tilde{\theta}$ is a vector of random parameters and X is the training set. However, in many important applications, this is infeasible.

An interesting question is thus how to compute the posterior distribution. Approximate Bayesian Computation is such an algorithm to compute the posterior distribution with the advantage that no explicit form of likelihood function is required. This is extremely useful in settings where the dimension of data is huge, but the underlying parameter has a relative small dimension.

The basic idea of ABC algorithm is very simple: one can use rejection sampling to obtain draws from the posterior distribution without computing any likelihoods. As shown in Algorithm 1, we draw parameter-data pairs (θ', X') from the prior $p(\theta)$ and the model $\mathcal{M}$, and accept only the θ' such that $||S(X') - S(X)|| < \varepsilon$, where X is the dataset we have.

Algorithm 1 ABC Rejection Algorithm

```
for i = 1, 2, ..., n do
    repeat
        Propose θ' ~ p(θ)
        Draw X' ~ M given θ'
    until ||S(X') − S(X)|| < ε
    Accept θ' and set θ^(i) = θ'
```

One of the key step in ABC algorithm is to judge whether the data X' generated by parameter θ' is close enough to the sample data X. When X is of high dimension, the probability $\mathbb{P}(||X - X'|| < \varepsilon)$ is very low (the sparsity of high dimensional space, i.e. the curse of dimensionality). So it is easy to go to extreme when choosing ε: either the rejection is of extreme inaccuracy, or accuracy at the expense of a very time-consuming procedure.[1]

So a central question is to construct a summary statistic S which can efficiently get most information in the data and has low dimension. With such a summary statistic, it is easy to get a good balance between rejection accuracy and computation efforts. In an abstract way, I want to use the posterior mean $\mathbb{E}(\theta|X)$ as a summary statistic, which has been proved to capture the first order information when summarizing X. However, we should note that although mean, median etc. are very commonly used, they can be very different from $\mathbb{E}(\theta|X)$.

So I want to apply some **automatic statistic selection methods** to learn the form of $\mathbb{E}(\theta|X)$ given the statistical model. To combine what I have learned in the course Machine Learning, I propose the following question:

How different machine learning methods perform

[1] When accuracy is high, i.e. ε is low, it takes many samples for enough acceptations.

for choosing the summary statistic?

Although ABC algorithm is well-known, the automatic generation of summary statistic is a recent research topic, and there are few papers on it. Blum (2010) and Fearnhead and Prangle (2012) use regression methods, and Jiang et al. (2015) use deep neural network. However, non of them compares the performance of different machine learning techniques in generating summary statistics. In this project, I aim to use different machine learning methods to generate summary statistics and compare the pros and cons of each method. This will provide guideline for the applications of this novel method.

II. Methods

I denote $\mathcal{M}$ as the statistical model, $\tilde{X} \in \mathbb{R}^p$ as the random values, and $X \in \mathbb{R}^p$ as data.[2] Moreover, to make the methods work, I assume that $\tilde{X} \sim p(\cdot|\theta)$, and it is possible to sample from this conditional distribution. Let $\mathcal{L} = \{$ generalized linear regressions, neutral network, $\cdots\}$ be the set of supervised learning algorithms, and $\ell \in \mathcal{L}$ to be an element of these algorithms. Then the main problem is to construct a low dimension and informative summary statistic S for high dimensional X. Steps are as follows. For each $\ell \in \mathcal{L}$,

- Generate a data set $\mathcal{D} = \{(\theta^{(i)}, X^{(i)}), 1 \leq i \leq n\}$, where $X^{(i)}$ is a random sample from the distribution $p(X|\tilde{\theta} = \theta^{(i)})$.
- Use the data set $\mathcal{D}$ to train the supervised learning model ℓ, to minimize the objective function

$$\frac{1}{n}\sum_{i=1}^{n} ||\theta^{(i)} - \hat{S}(X^{(i)})||_2^2 \tag{1}$$

  This objective function can be viewed as an approximation of the posterior mean $\mathbb{E}(\theta|X)$ because the solution to an squared error minimization problem will results in such an estimation.
- Run the ABC algorithm with the summary statistic $\hat{S}(\cdot)$ to get a posterior distribution. Then check the calculated posterior distribution with the posterior distribution derived from the sufficient statistics to judge whether the machine learning method ℓ performs well .[3]

There are two sources of potential errors: One is from the construction of statistics, where $\hat{S}(\cdot)$ is not necessarily a sufficient statistic so that there is some information loss. The other is from the ABC algorithm, which contains errors because of the acceptation criteria $||\tilde{X}(\theta) - X|| < \varepsilon$. To guarantee that there is indeed a meaninful comparison among different machine learning algorithms, I need some convergence results for the estimation methods. The following theorem is from Jiang et al. (2015).

Theorem 1. *Assume that $\mathbb{E}|\theta| < \infty$. ABC procedure with observed data X, summary statistic S, any finite vector space norm $||\cdot||$, and tolerance ε produces a posterior distribution $p^{\varepsilon}(\theta)$. Then we have*

$$||\mathbb{E}_{p^{\varepsilon}}[\theta] - S(X)|| < \varepsilon$$

and

$$\lim_{\varepsilon \to 0} \mathbb{E}_{p^{\varepsilon}}[\theta] = \mathbb{E}[\theta|X]$$

This theorem guarantees that when ε is small enough, the error from ABC algorithm can be neglected, and thus the evaluation of different supervised learning algorithms is valid.

Moreover, we can use ABC algorithm to get $E[g(\theta)|X]$ for any continuous functions $g(\cdot)$, and thus get the whole posterior distribution. Because the objective of this project is to evaluate different learning methods, it is easier if $E[g(\theta)|X]$ is sufficient for the posterior distribution. So I use a one dimension θ with a distribution that only has $E[\theta] = \mu$ as the parameter. This is also the reason why I use the quadratic objective function.

III. The Model for Evaluation

In this project, I have to choose a statistical model $p(\cdot|\theta)$ to evaluate different methods. The following property is needed for evaluation.

[2] In typical applications of ABC algorithm, p is very large, e.g. $p = 200$.

[3] From the purpose of this project, I have to use models where the sufficient statistic is known.

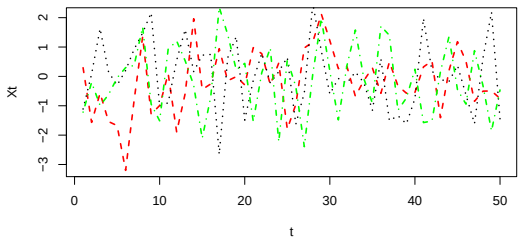


Fig. 1. Typical MA(2) process with $(\theta, \gamma) = (0.5, -0.5)$

- The dimension of parameter θ is small, but the dimension of data is very large.

I will use different machine learning methods to get a summary statistic and compare the performance.

A. Moving Average model

Let $X \in \mathbb{R}^p$ be a vector of observations in an $AR(q)$ time series. Assume the underlying model is

$$X_i = \varepsilon_i + \theta\varepsilon_{i-1} + \gamma\varepsilon_{i-2}$$

where Z_j's are unobserved random noise terms. To get an analytical formula for the likelihood function, I assume that $\varepsilon_j \sim N(0,1)$ and are i.i.d, and the true parameter is $\theta = 1, \gamma = -1$. Assume the prior is a two dimension standard normal distribution, so that I can compare the posterior directly with the true distribution. Typical MA(2) processes with $\theta = \gamma = (0.5, -0.5)$ are shown in Figure 1.

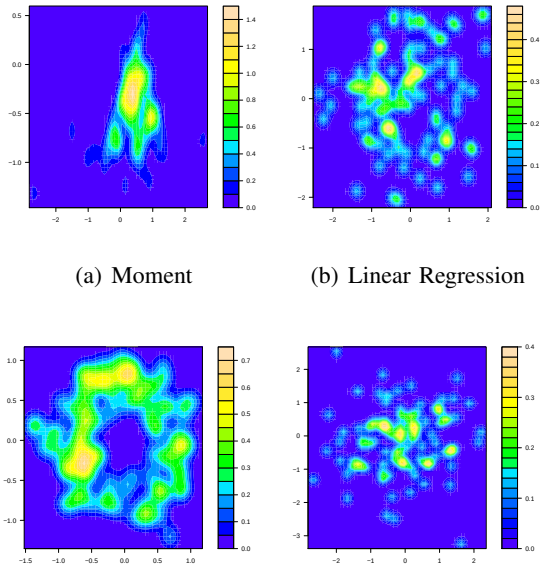
Then I compare the following methods:

- A natural moment estimation

$$\hat{\theta}_1 = \frac{1}{n-1}\sum_{i=1}^{n-1} X_{i+1}X_i$$

- Using linear regression model.
- Using Radial basis function network.
- Using Neural network.

I simulate the MA(1) process with total time length $T_0 = 100$, and replication number $N = 1000$. The observation is one sample from the true parameter. Then I separately use the above methods to estimate the statistic and posterior distribution, as shown in Table I and Figure 2.



(a) Moment (b) Linear Regression

(c) RBF (d) Neural Network

Fig. 2. Posterior Estimation for the Time Series Model

I find that there are significant differences among the four methods. Because the prior has mean $(0, 0)$ and the true value is $(0.5, -0.5)$, a good updating should be somewhere between the above two values. Based on this criteria, I find that the simple moment summary statistics performs the best. This shows that if we have some summary statistics with nice properties in theory, it is better than other learning algorithms. This can be seen from the heat map in Figure 2(a), where the posterior is centered around, while other methods produce a scattered posterior. Moreover, this also results in lower standard deviation. Comparing Figure 2(b) (c) and (d), I find that linear regression produces a very scattered posterior, while neural network produces a more centered posterior. Moreover, the RBF result is even more scattered. This suggests that neural network performs better.

B. Hidden Markov Chain Model

I use a bistable system that can be characterized by a hidden Markov model (HMM) subject to measurement noise, as illustrated in Figure 3. There are two states: 0 and 1. The probability of a transition from one state to the other is defined as θ in both

TABLE I
POSTERIOR SUMMARY STATISTICS FOR THE TIME SERIES MODEL

Posterior Statistic	Moment	LinearRegression	RBF	Neural Network
mean(θ)	0.394	-0.113	-0.099	-0.096
mean(γ)	-0.465	0.043	0.041	-0.131
std(θ)	0.604	0.961	0.626	1.049
std(γ)	0.348	0.942	0.577	0.966
corr(θ, γ)	0.245	0.100	-0.039	0.071

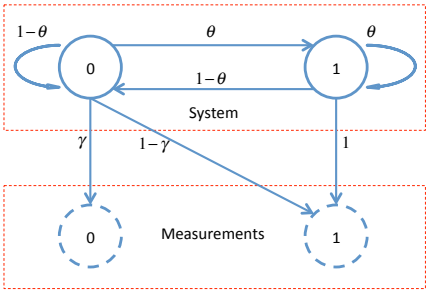


Fig. 3. An illustration of the hidden markov chain

directions, and the probability to remain in the same state at each time step is $1-\theta$. Moreover, the probability to measure the state correctly is γ (So the probability of an incorrect measurement is $1-\gamma$). It is easy to get the stationary distribution as

$$\pi(0) = 1-\theta, \quad \pi(1) = \theta; \quad \tilde{\pi}(0) = \gamma(1-\theta), \quad \pi(1) = \theta+(1-$$

Assume the starting state is always 1. Note that we cannot estimate both γ and θ by a simple moment estimation, because in stationary state they are unidentifiable. We can only use the non stationary sequence to infer both of them.

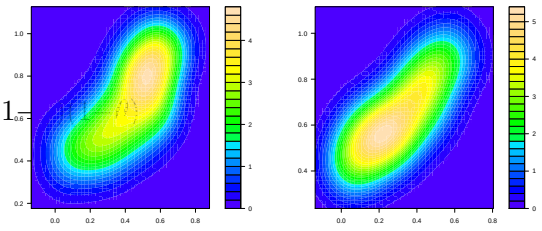
Let the prior distribution be two independent $U(0,1)$, and the true values be $(\theta, \gamma) = (0.5, 0.9)$. I will compare the following models.

- Linear regression
- Radial basis function network.
- Neural network.

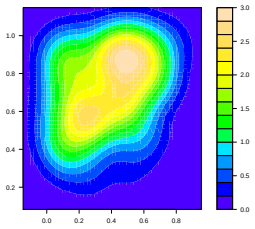
The results are shown in Table II and Figure 4. I find that the results are very similar, except that the standard deviation is slightly smaller for linear regression.

TABLE II
POSTERIOR SUMMARY STATISTICS FOR THE HIDDEN MARKOV CHAIN MODEL

Posterior Statistic	LinearRegression	RBF	Neural Network
mean(θ)	0.512	0.520	0.508
mean(γ)	0.512	0.520	0.508
std(θ)	0.268	0.271	0.279
std(γ)	0.268	0.271	0.279
corr(θ, γ)	0.866	0.905	0.730



(a) Linear Regression (b) RBF



(c) Neural Network

Fig. 4. Posterior Estimation for the Hidden Markov Chain Model

IV. Discussions

In this project, I propose methods for learning summary statistics via approximate Bayesian computation, including Radial basis function network, neural network, and some simple methods like linear regression and moment condition as a benchmark. The theorem that ABC algorithm asymptotically approaches the posterior guarantees the validity of the method.

To evaluate, I designed two example: one is the typical time series model, and the other is a hidden markov chain model. Both of them have the feature that the dimension of the data is much higher than the dimension of the parameter. This is the typical application of ABC algorithm. Then I use different methods to learn the summary statistics and compare their performance. I find that a simple moment estimation outperforms the more complex methods. Moreover, neural network produces a more robust result.

The drawback of the current approach is that I cannot accurately evaluate the posterior as I don't have an analytical form of it. A better way to evaluate different methods would be doing ABC updating for multiple samples, instead of on one sample. Using the property that posterior converges to the true distribution when the sample size is large, I can measure the posterior from ABC with the true distribution as a goodness indication. However, currently I don't have a reliable way to do multiple updating accurately using the ABC algorithm, as the rejection sampling method loss a great fraction of data in each updating. Probably I can use non-rejection sampling method for the updating part, but that will be a different algorithm than ABC.

References

Blum, M. G. (2010). Choosing the summary statistics and the acceptance rate in approximate bayesian computation. In *Proceedings of COMPSTAT'2010*, pages 47–56. Springer.

Fearnhead, P. and Prangle, D. (2012). Constructing summary statistics for approximate bayesian computation: semi-automatic approximate bayesian computation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(3):419–474.

Jiang, B., Wu, T.-y., Zheng, C., and Wong, W. H. (2015). Learning summary statistic for approximate bayesian computation via deep neural network. *arXiv preprint arXiv:1510.02175*.