Simulated Annealing used to fit Lotka–Volterra equations

Jasper den Duijf
Koen Greuell

Abstract

Highly dimensional models are hard to fit to data points as it quickly becomes computationally too expensive to explore all possibilities within the whole sample space and also impossible to solve mathematically. Lotka-Volterra is one such an example of a multidimensional problem. It can be used to describe the interaction between predators and prey. Simulated annealing is a stochastic strategy to randomly select possible input parameters for a model in order to approach the optimal fit. We will use simulated annealing in this paper to try and fit the Lotka-Volterra to data, thereby comparing different possibilities of settings of the simulated annealing algorithm and comparing it to a hill ascent strategy. We find that the hill ascent strategy results in a better fit to the data than the simulated annealing method we developed.

Contents

1 Introduction

When prey and predator are observed in a territory, quite often we observe a certain cyclistic behaviour. The predators eat the prey, declining the preys population. This causes the predators to starve, declining the predators population. With a lack of predators, the prey's population can grow and finally with enough prey the predators population can grow. This behaviour is described with the Lotka-Volterra equations, where the speed of the growth of the populations depend on the size of said populations. Beside that the model requires four parameters which control the growth and decline of the populations.
Now for certain data that we believe to follow this pattern, we try to find the parameters of the model. For models with a high amount of input parameters however, it is not possible to calculate an optimal fit to data by simply using a least squares algorithm, also due to the complexity of the equations. As the multiple inputs create a multidimensional sampling space it becomes quickly impossible to explore the whole space and a strategy must be applied to stochastically explore the space. One of such techniques is simulated annealing, which was inspired by metallurgy, in which the molecules in a metal stabilize in a more stable form when they cool down slowly [1]. In a similar way the space can be explored, slowly cooling the process down as an optimum is found. We will try to find an optimum fit of Lotka-Volterra equations to predator prey data to see whether simulated annealing is a good technique to realize a fit in multiple parameter models. We'll compare this with other algorithms, like the Hill Climber algorithm and vary the properties of the simulated annealing like the temperature and cooling down schedule, to answer the question how to use this algorithm.
In chapter 2 we will explain the theory behind the Lotka-Voltera equations and show those equations. We'll briefly discuss Markov chains and the mechanisms of the simulated annealing algorithm. In chapter 3 we will digress on the methods used in our simulated annealing algorithm in more detail, evaluating certain properties. In chapter 4 we'll explain here the differences between the algorithms and compare the results. In chapter 5 we'll discuss this paper and evaluate our research.

2 Theory

2.1 Lotka-Volterra equations

The Lotka-Volterra equations are first-order non-linear differential equations, firstly described in 1910 [3]. These are equations existing of derivative of unknown functions and the products of those functions. Given a certain starting point their state will change over time. The term first-order implies that only the first derivative is used in the equation set. We see the general

form:

$$\frac{dx}{dt} = \alpha x - \beta xy$$
$$\frac{dy}{dt} = \delta xy - \gamma y$$

In this equation x stands for the amount of prey and y stand for the amount of predators (this is why the Lotka-Volterra equations are quite often called the predator prey model). To give the parameters meaning, α can be considered the birth rate of the prey. In the first equations that with an $\alpha > 0$ the amount of prey grows faster when there is are lot of prey. β can be considered the catch rate. The population of prey declines faster with a higher catch rate, however this is also very dependant on both the amount of predators and prey. γ is the death rate of the predators: the more predators are in a territory, the higher this parameter and the amount of predators will drop faster. This is nullified by δ, the birth rate of the predators, which not only depends on the number of predators but also on prey.
The population of both species at time t can be considered as coordinates, like on time t_0 the system holds (x_0, y_0) species. An interesting property of this set of equations is the population equilibrium, where the system does not change anymore. We can derive two fixed points, either $(0, 0)$ and $(\frac{\alpha}{\beta}, \frac{\gamma}{\delta})$.
The general pattern of the system looks like two sinusoids out of phase, as can been seen in figures in the results.

2.2 Markov chain Monte Carlo

Simulated annealing is a Training-based Markov chain Monte Carlo algorithm. It is one of the uses of the Monte Carlo methods, which tend to follow the same pattern: firstly a domain for the inputs is set, then a number of random inputs is created in this domain and eventually in a deterministic way those inputs are used to calculate a result. It is broadly used for computations that are impossible or computationally too expensive to do deterministic, like predicting or optimizing complicated models or integrate complex functions.
A Markov chain is a series of states and their underlying iteration wise changes from one state to another. One of the most important properties of a Markov Chain is that is does not matter how one reaches a certain state, the probabilities of the next states are independent of the previous states with exception of the current state. Eventually in the limit the state of the system goes to a distribution of a sort and this can be used to analyze the system.
Together a Markov chain Monte Carlo is a system that goes through a various amount of states, eventually either passing the state that one requires ór passing through enough states to say something about the distribution of certain properties.

2.3 Hill Climber

A hill climber algorithm is an optimization algorithm based on a random starting point and local optimums. All points are based on a combination of parameters within a given domain. In short, for every point it finds a neighbour point that is better, repeating itself several times until no local improvements are possible. Thus it is able to find local optimums. Whether these optimums are just local or perhaps global depends on the starting point. Usually this algorithm is done many times consecutively with different starting values to find multiple optimums. If this is done for infinite times, logic dictates that eventually one of the local optimums has to be the global optimum.

2.4 Simulated Annealing

Simulated annealing has certain similarities with the Hill Climber algorithm: it also starts with a starting point and checks whether a change in the parameters gives an improvement. However even if a reduction occurs, the change might be accepted. This depends on temperature, a parameter that decreases with the number of iterations in the algorithm. It also depends on the how much the goal value diminishes. One of the ways to do so, which will be used in this paper, is a Boltzmann distribution. This says given a point x_i^* with a minimizable goal value E_i^* and a the next point x_i with E_i is always accepted if $E_i < E_i^*$ or with a chance:

$$P(x_i^* \to x_i) = e^{\frac{-E_i^* - E_i}{kT}}$$

here is k the Boltzmann constant. We see that whenever the new E_i gets bigger in comparison to E_i, the chance decreases. Also we see that a lower temperature causes a decrease of this chance. During this paper we'll search for a reasonable cooling down schedule for T.

3 Methods

3.1 Domain

To decide the domain of the input parameters used for the Lotka-Volterra equations, we use the theory that all parameters are always non-negative numbers and use x_k, y_k and x_{k+1}, y_{k+1} as indicators. Since $x_{k+1} \approx x_k + h * \frac{dx}{dt}$ as we follow the Euler approach, we can estimate $\frac{dx}{dt} \approx \frac{x_{k+1} - x_k}{h}$. In the similar way we can do this for $\frac{dy}{dt}$. All our parameters $\alpha, \beta, \gamma, \delta$ appear to fall in the domain $[0, 10]^4$, which is used to pull random variables from.

3.2 Initiation

An initial fit to the data was made using a random values for the parameters within the range (0,10] and the line was fitted using integration from ODEPACK [2]. For the time points where

data was available the mean squared error of the difference between model and data was taken.

3.3 Simulation

A Markov chain was created by randomly changing each of the variables to a new value within the range with a probability of 0.5. If none of the variables was selected to be changed the selection was repeated. After selection of new variables the model was rerun and the mean squared error (MSE) of the new fit was determined. For simulated annealing algorithms the new variables were accepted if the mean squared error was either lower or

$$e^{(-\Delta sum\ of\ squares/T)} > u$$

where the random number u was selected from a uniform distribution in the domain (0,1]. This mentioned acceptance of parameters with a higher of squares we refer to as tolerance and decreases with the temperature. After a predetermined time the Markov chain was closed and a new Markov chain initiated. The temperature (T) decreased after each Markov chain following a pattern of exponential decay (figure 1. After a predetermined amount of Markov chains the program stopped and the mean squared error of the model was compared between different settings.
The model was run with 10000 iterations and Markov chains of length 100.

3.4 Cooling schedule

One of the most important decisions is with which cooling schedules and number of iterations to run the simulation. We require the cooling schedule with the least entropy, that will lead to the global optimum with the highest probability. To find this we compare a few cooling schedules with different temperatures. Since the real global optimum is unknown, we can compare the probability of a new point being accepted. We would prefer this chance to be high at the beginning and slowly go down to 0.
We can either change the temperature after every new considered point, or keep the temperature steady for a number of iterations, creating a Markov chain with multiple iterations the same temperature. We can see the different cooling schedules in (figure 1), where we see that both of them drop discretely exponentially to 0. We can verify for the decrease of the mean squared errors that the temperature decrease indeed causes a lower probability of acceptance.

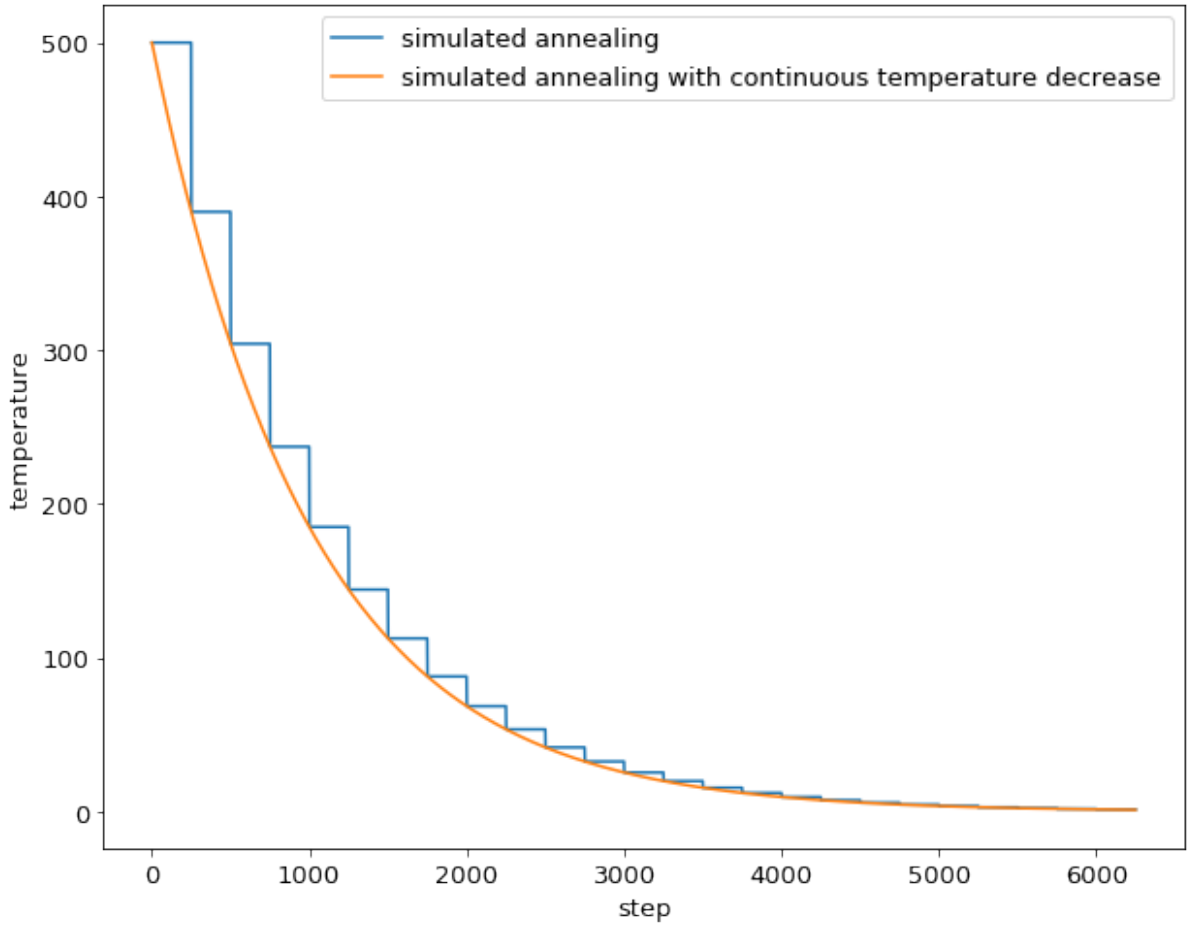

Figure 1: Two possible cooling down schedules.

3.5 Comparison

To make a fair comparison between different methods, each method was run for an equal amount of iterations. For the hill climber algorithm this meant that it was repeated 25 times from different random starting points.

4 Results

4.1 Simulated annealing

We ran the simulated annealing algorithm several times, every time with 10000 iterations. We noticed quite a lot of variation in the result. To illustrate this and to check whether the algorithm preformed properly, we plotted of a couple of results how they changed over time (figure 2). Since we can not properly display the four-dimensional space that is the parameters setting, we have placed the mean squared error against the number of iterations and plotted this, as

shown in 2. Although we notice a chaotic amount of sudden peaks, we can also see that the great outliers disappear near the end.

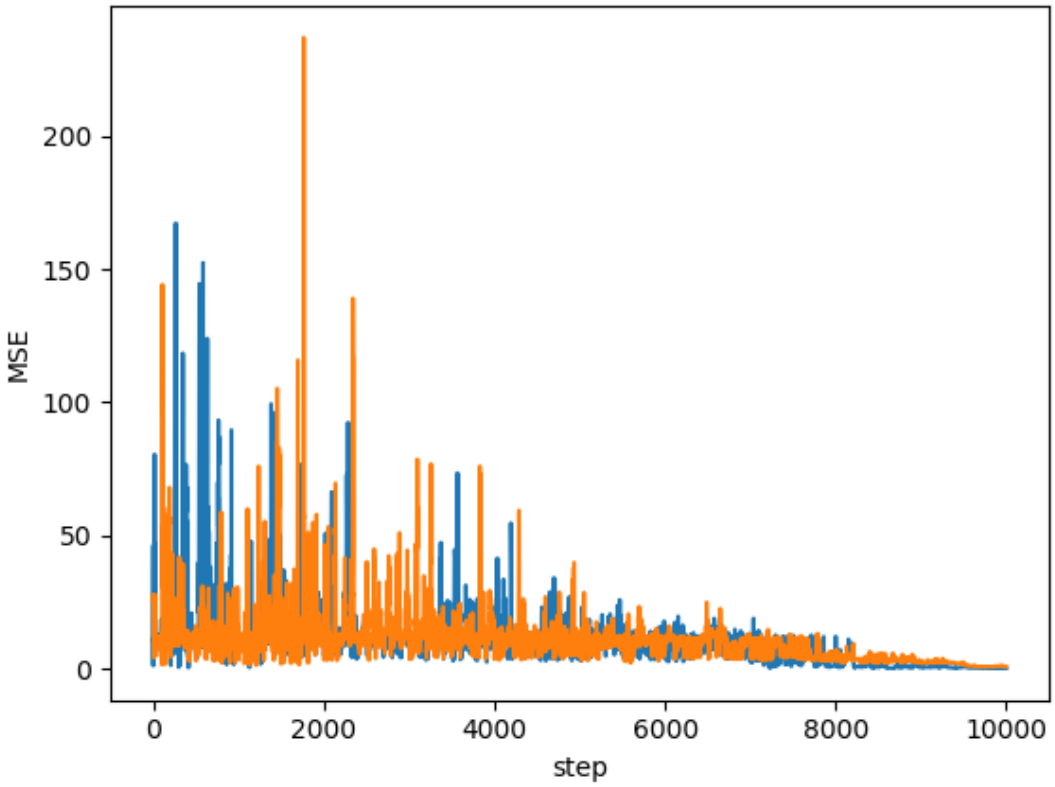

Figure 2: In this figure the path of two simulated annealing simulations are plotted. We see that it does not always keep finding better answers, but is allowed to make errors, thus accepting for a short time higher MSE's. However as the number of iterations increases the temperature get lower and thus less larger error will occur, eventually flattening out.

The best result found with the simulated annealing algorithm was with resulted in values of α,β, δ and γ of 1.04, 0.49, 1.59, 0.93. The mean squared error at these values was 0.073. However if we look at all the outcomes of the simulated annealing we notice quite some variation in both parameters and parameter settings. It was computational to expensive to draw significant conclusions about the mean of these results or the standard deviation, for that we could simply not create enough simulated solutions. We're absolutely not capable to call this result the global optimum.

4.2 Hill climber

For the hill climb algorithm initiated at 25 random points the optimum after 250 perturbations was found at an α,β, δ and γ of 0.71, 0.38, 2.4, 1.3. This gave a mean squared error of 0.047 (figure 3). This fit is considerably better than the fit found by simulated annealing.

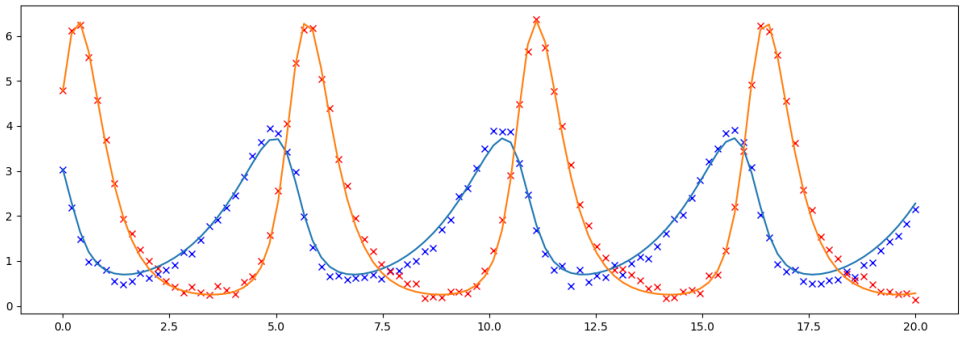

Figure 3: Hill climbing fit to the data.

4.3 Lowest mean squared error strategy

When the algorithm was set to take the parameters delivering the lowest mean squared error from the Markov chain to the beginning of the next Markov chain, the established parameters α, β, δ and γ were 1.22, 0.652, 1.52, 1.03, delivering a final mean squared error of 0.1976. The lowest mean squared error strategy performs equally well as the simulated annealing where the last parameters in each Markov chain are used as input to the new chain.

4.4 Continuous temperature decay

When the algorithm decreased the temperature after each iteration rather than after each Markov chain, the parameters α, β, δ and γ found were 1.61, 3.42, 4.07 and 6.82 and the final mean squared error 10.31. The continuous temperature decrease strategy therefore performs much worse than simulated annealing with Markov chains where the temperature is kept constant during the Markov chain.

5 Discussion

In this paper we used a couple of methods to fit data in an optimal way to a set of Lotka-Volterra equations. To do so we try to locate a set of parameters that will yield the best fit, measured by the mean squared error. We have made use of multiple variations on a simulated annealing algorithm, as well as a hill climber algorithm.If the computational time is limited or only an

approximate solution is required hill descent from multiple random starting points may be the fastest technique to obtain an approximation of the best solution.

The performance of the model seems to be highly dependent on the tuning of the exponential decay of the temperature. A too high starting temperature causes a waste of iterations on accepting too many new states and a not low enough final temperature allows the model to accept new states which are worse than the previous. We did multiple trials to find a good the starting height of the temperature. We noticed nearly no sustainable differences, so we put this value on 500. However even more importantly than a starting value is the ending value. During our research we had this value put on 1. However upon close inspection with a T of 1 and a Δ mean squared error of 1, this deterioration was still accepted with a probability of $\frac{1}{e} \approx 37\%$. When we lowered this value to 0.1, we got improved results. We recommend this for further research.

We had foreknowledge of the data, as we knew that the data were derived from Lotka-Voltera equations with Gaussian noise added. This would mean that it is not possible to create a perfect fit, but our hypothesis upfront was that a solution should exist with a low MSE and that the algorithms should be able to find those points. However the results did not meet those expectations while verifying the model. We experienced some technical errors in the code of the algorithms and we were never able to fine tune the results. Eventually we realized that the algorithm was not properly implemented, which might have had something to do with the following problems.

There are a few results that were different then expected. For example we have plotted the lowest MSE's as the temperature drops. We would expect in the beginning with high temperature that these errors were high, since the algorithm would be able to roam freely through the parameter domain. As the temperature dropped, the error would be lower, pushing the solution to an local optimum. However there is no significant change of the MSE when we reduce the temperature.

One of the things that we could change is the starting point. We now let the model always begin on the exact start of the data. We know however that this might not be the case. For further research we would suggest rerunning the code with starting point $(x_0 + x_\psi, y_0 + y_\psi)$ where x_ψ and y_ψ will be the distribution between the first point of the model en the first point of the data, not necessary equal to 0. We could adjust this value similar like we did with parameters α e.d, although this would add two dimensions to the input of the model and thus makes it extra difficult to solve. Because this was computational too expensive at this point we opted out of this possibility.

Other experiments that we were not able to do was to check whether the data was still fittable with less data points, or analyze the importance of the starting value. We also would like to analyze the importance of the definition of a "local space" in witch the changes are allowed. We now have this limited to 0.5 for every parameter. When this gets lower this will immediately

effect the effect of the hill climber algorithm, as now the starting point becomes more important. For the simulated annealing algorithm we expect this to be less interesting, but more research on this topic is necessary.

To conclude we would address the low amount of statistical substantiation, as we ran into many technical difficulties, which costed too much time. Certain conclusions drawn in this paper, like the fact that the Hill climber gives better solutions than the simulated annealing, can therefore not statistically substantiated.

References

[1] Azencott, R. (Ed.). (1992). Simulated annealing: parallelization techniques (Vol. 27). Wiley-Interscience.

[2] A. C. Hindmarsh, "ODEPACK, A Systematized Collection of ODE Solvers," in Scientific Computing, R. S. iterationleman et al. (eds.), North-Holland, Amsterdam, 1983 (vol. 1 of IMACS Transactions on Scientific Computation), pp. 55-64.

[3] Lotka, A. J. (1910). "Contribution to the Theory of Periodic Reaction". J. Phys. Chem. 14 (3): 271–274. doi:10.1021/j150111a004.