Master Thesis

Applying Machine Learning For Generating Synthetic Geospatial Trajectories

BY
Yannick Patschke

Prof. Benoît Garbinato
PhD Candidate Vaibhav Kulkarni

Abstract

Applying machine learning for generating synthetic geospatial trajectories. Learning mobility behaviors of entities by applying recurrent neural networks (RNN) utilizing the learn models to generate synthetic traffic.

Contents

1 Introduction

From the begins of the humanity, humans have always needed and tried to adapt to their environment and new environments. We have also rapidly understood that we were lazy.

So since then we have done all our best to find a way to reduce our effort and maximize our results. The first men have tried to use tools to simplified their life and survive; we can think of the use of mace, knife, spear, torch and so one, rapidly not only just for protection, but also to cook, light up or other reasons.

Then when the survical among other species was no more our main problem, we have begun to think larger, and tried really to increase our results in every part of our life; by beginning to construct the first villages and cities, to regroup the production, to use tools to plow our fields. Then after that we have even replace men by machine when it was possible. We have tried to automatize the most possible tasks of our society.

Finally with the arrival of computers and the internet, we have increased even more this automation of things. So it isn't anymore a secret that technologies, especially those related to computers, have become essential to the proper functioning of our society.

Now we have begun not only to replace, but also really trust the computer and its power, to do stuff that we will not able to archive or only very slowly. And even more we are at a point where the machine can begin to not only just do what we have explicitly said to do, but begin to do stuff or learn without being explicitly programmed. They can do things and adapt them self following the context in which they evolue, and that it is: **Machine Learning**.

1.1 Machine Learning

Machine learning: a big term, but what is it exactly ? What can we do with it ? Where does it come from ? Why so widespread ? Machine learning is basically, to teach to the machine to act in situations that differ from what we have shown or trained her.

It's a process where we train the machine with examples and then give new but similar data and let the machine try to work with this new data based on what it has learn previously. Without going into details, now a days we distinguish two major types of machine learning; supervised and unsupervised.

Supervised machine learning allows us to train the machine based on example data and their results. So basically during supervised machine learning, we show the machine our examples and their correct results several times. Until the machine have learn some pattern about the data, and understand that if we give her a input A the correct answer will be B. For instance, we can make a connection with how a human kid learns to read. The kid will try to read some simple books a several times, until he correctly reads and articulates all words of these books. So when he has correctly read all these books entirely, we can assum that he understands how read different words, and we can assume that if we give him another book, he will manage to read this new book even if it's the first time. Fot the machine, it's similar; each time the machine sees the data example, she will try to remember what she has done good or bad and try to make better the next time. So we can say that supervised machine learning, it manly use to understand the way to go from a point A (our input data) and a point B (our output data, the correct answer).

Unsupervised machine learning works differently; this time the machine will just train on examples without feedback of what is true or false, it means without correct answers to find of the example data. So imagine again a child who is given a pile of marble to tidy up separately, but with no other instruction about criterias to separte them. Maybe the child will seperate them by color, size, weights or others, maybe mixed criterias. But he will try to find a way; the machine does the same. The machine will for example try to identify some attributs about the data that are similar between some examples and try to find a way to group them together. So here it's an example about clustering and we can see that unsupervised machine learning is more to find some structure in the data given.

But where and when we can use unsupervised or supervised machine leanring ? Nearly everywhere and any times, if we are attentive. In our society, we can find traces of machine learning everywhere. For instance in our email with the spam classification, with some social network with facial detection on pictures, with speech recognition for example from Siri of Apple., and so many other examples can be found, if we search a little bit.

1.2 Motivation

1.2.1 Mobility Traffic

Why work on mobility traffic data ? What are the utilities ? We can answer to that by speaking about the increase of movements. Not only human movements, but also materials or informations, the number of shifting and movement in our society has never been so high in our history, and continues to increase.

For more than a century, we have not stopped seeing advances in many fields; mechanics, medical, computer science and many other sciences. Every time, these advances are reserved for a small number of people, but rapidly popularized and accessible to all. We can think of first cars rapidly pupularized with the *Ford T* by the brilliant *Henry Ford*.

Especially since globalization, all new things have become even be more rapidely accessible for the general public. So a lot of things that were one century ago reseved to rich or a particulary sector, for instance travel, are now common and widely widespread.

So our technology and also our way of life, now days have particulary influenced movements in our society. They have really increased and promoted increasing movements on people, informations and materials.

It's really difficult to imagine the amount of data that we could collect just about a movement of goods in one day in a city, and how much about the number of letter ships or people moving to work. Getting all these informations is possible at the technological level, now that some technologies as gps and connected device are so widely used and accessible. All these different datas about mobility can be usefull in some ways. If we think only about human mobility, we can already think about a lot of fields where this data could be useful:

Traffic management: We can analyse the movement of people to better know the most affected areas and think of a way to improve the fuidity or prevent of traffic jam.

Urban planning: With behavior of movement of people, we can predict the best areas to construct, or maybe think how to increase existing area level.

Consumer profiling: We can try to learn behaviors of consumers to better understand theirs needs or find a better way to reach them.

1.2.2 Generating Synthetic Geospatial Trajectories

So now that we have seen that Mobility Traffic data can be useful in different purposes, why do we need generating synthetic mobility traffic ? Why not just collect directly real data in the real world ?

We need to understand that for all the above mentioned fields on mobility traffic, we really need a lot of data to obtain results which could make sens. It's not enough to just go and ask to 100 people to give use their mobitlit data.

So why not just take all data that we need ? The main problem is to collect all this data. As said before at a technological level we could obtain nearly all this data, but in practice we are stopped by the amount of data. It's really difficult for somebody, a city, or an organisation to obtain enough data. Because you need to ask people directly, which means really a lot of people.

The second main problem will result of the will of people. Not a lot of people will aggree to give freely their mobility data. So there will always be the barrier of privacy. Rare are those who accept to give all their movements in life.

And thinking of paying people to give us them mobility traffic data is just not imaginable. If it was the case, people would maybe participe, but the city or the organisations would need really a lot of funds.

So why not force them to give us the data or even find a way to collect their data without their agreement ? It would be possible, only if we where in country without a sense of democracy or at a time not so distant when democracy had not really emerged.

So for all the above reasons, generating synthetic geospatial trajectories could be an interesting thing in the end,if we needed less data, whcih would mean less problems about collecting data.

1.2.3 Random Walk Model

So as said, generating data could be usefull, but why speak about machine learning for that task, why not use an easier model as the Random Walk Model to generate data ? Yes, a random walk model can be imagined and applied in our case.

The random walk model is a simple process that could easily and fast be implemented. It's a process where the current value of a variable is composed of the past value plus an error term. The implication of this model is that the change of y, so in our case the next position, is absolutely random.

So the use of this model to generate movement will have results, but what kind of results ? It will be simply movement totaly random, but people and our society work and think not in a random way, so this model will not capture reality. It will not capture the real behaviour of users. It will not look like a real human movement. And having human movement data is stricly needed for projects working on mobiltiy traffic of people. So we can rapidly disaprouve and forget a simple model as the random walk model for our task.

Example of eight random walks in one dimension starting at 0.

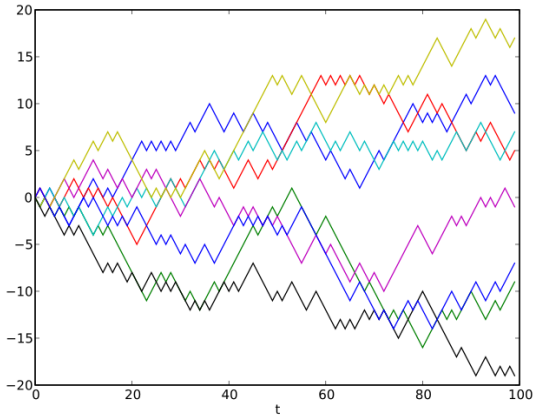

Figure 1: Source: Wikipedia

It's this problem of capturing human behavior where machine learning could be usefull. In other complexe fields, machine lerning has already given surprising good results. So why not try in our case and see if the results are interesting.

2 Literature

In the context of machine learning, cases and studies are very wide and numerous. Other people have already worked on a similar or at least a project with some concept in common with mine. So trying to discover and work all by myself without according regards to the works of others would be like wasting time.

In our socity where globalisation, interconnected interest and sharing competences have proved their rentability, I couldn't imagine not benefitting from knowledge of others. So I have passed trough some papers, project, discussion or book of others in order to find some useful help for my work.

I was rapidely overwhelmed by informations about machine learning, but I fastly needed to sort. It was obvious that some parts could help me to understand more my work and future work or at least help me to have different idees or take a step back on my subject. So in this section are several helpful things that helped me during my project.

1. A Coursera cours named *"Machine Learning"* from Stanford University teached by *Andrew Ng* which teach me some basic understanding about machine learning and its different possibilities.
2. The article of *Andrej Karpathy* on *"The Unreasonable Effectiveness of Recurrent Neural Networks"* [13] which provide some very good explanation about Recurrent Neural Network.
3. The library *TensorFlow* that allows me to work with machine learning in a efficient way with python.
4. The github project of *Aymeric Damien* on TensorFlow examples that provides me some basic examples of different model of machine learning using TensorFlow.

5. The article of *Vaibhav Kulkarni and Benoît Garbinato* on *Generating Synthetic Mobility Traffiic Using RNNs* [14] that provides me some base one the subject of Synthetic Mobility traffic using Recurrent Neural Networs, some motivations about generating them and some issues linked.

6. The book *"The Elements of Statistical Learning"* reinforced some previous concept, particuliary the ones learns from the Coursera course.

Obviously there was much more documents, articles, blogs, forums, tutorials and so on that accompanied me throughout my project, but these were the most decisive in the good conduct of my work on the subject of Applying machine learning for generating synthetic geospatial trajectories.

3 Approach

Every project need to have a plan to follow or to try to follow. So to archive my project of creating generating synthetic geospatial trajectories, I divided it in three big phases: Predict, Optimize, Generate.

3.1 Predict

The first and important part of the project is to find a model that predicts the trajectories of users based on their previous position(s). For this we need to check and find a model that performs well and better than a classic model as logistic, linear model or random model.

We need to try, execute and compare different models. See how they perform on datasets we have available, check where and maybe why they perform better than others. When we are confident in a model and its performances, we can go to the next step to go further.

3.2 Optimize

After having selected a model on which work, the next step is now to begin to optimize and deepen the model selected. Begin to switch and compare parameters.

We need to find an adequate learning rate, a number of epochs during the training (number of time that our train data will pass in our model), number of neurones or the number of hidden layer for a neural network, and so one. We need also to check the best way to give him data, and try to go to the limit of the model with the datasets in our disposition.

So a big part of testing, switching some little things, compare them, and try to reach to conclusions and some results.

3.3 Generate

The last part of the project is about generating data based on our model. For example by giving our program a starting position and letting the model give us the next positions based on how it's predicted it. It is a

phase of generating and evaluating the data generated to see if it fits with what we expect and if it seems near the human behaviour observed in our datasets.

4 Data

For the project, we have access to two datasets: one from Lyon and one from Nokia. Each of them represent the position in the time of some users in their everyday life. A lot of other informations were collected in these datasets and are accessible for us, as for instance message sends, calls, battery level and so one.

4.1 Nokia dataset

I selected the nokia data, not only because it is more complete, but especially more interesting to me with datas about positions of my country and near my region.

This dataset contains 150 users that received smartphones to use and keep all time from 2009 to 2011. Smartphones have collected all data possible, as sms, phone call, network and many more data. It was always done preserving the confidentiality of the users.

But data that concerns us in priority was geolocation positions of the users in time. So basically the longitude, latitude and time per user. And with this data we can begin to use them and work on our subject.

So after having looked at the datasets, I needed to select a way to begin. Starting directly with positions (longitude, latitude) and time with machine learning is not necessary the best way; it can be really complex and has bad results. Especially, if like me, this is the first time that we work on machine learning.

So I have first simplified the data, by taking only what we call points of interest. These are only some areas on a map where users stay more than a certain time. So the data on which I work now is simply movements between these points of interest. So all this data are put in a sequence. We have then a sequence of index, where each index represents one of these zones where users stay more than a certain time.

For instance, the following sequence [0 1 0 2 3 ....] will represent the movement of one user. First, he is in the area 0 then goes to 1. Next he returns to 0, then goes to 2, and so one.

In further work we could imaginel maybe reuse the first data (longitude, latitude, and time, and also maybe more feature if we think that can help our case) and try to perform better and have also better results.

5 Prediction

5.1 Data preparation

As said in the section *Data*, I have worked on a simplification of the problem with sequences of point of interest (that we called POIs). So the first thing was to extract all transitions from users in shape of sequences of POIs. So having all data as [0 1 0 2 3 ....]

Then to rework them to have an easier and better shape to work on machine learning, so according with some readings, I choose to represent all positions by a vector binary which represents all POIs possible. We obtain a vector of a shape (1,238) because we identify 238 different areas where users stay more than a certain time. So a position can be represented by the vector of 238 binaries where only one of them will be equal to 1, and will be the index of the position listed.

5.2 Models

Now lets speak really about machine learning these last decades have allowed really to see the power of computer increase. So it allows to process a large amount of data faster than before. It makes more interesting to use machine learning; indeed machine learning needs often to work with a large amount of train data to be efficient. So these last years have been really good for this fields. We have seen more and more people using machine learning, not only in big organisations but also private people.

Indeed everybody doesn't need to have an expensive computer to make machine learning, but most of personal computers can now handle machine learning in a great way. Moreover a lot of courses have been created on the subject and a lot of services allow help us on machine learning, for instance: Tensorflow, Keras, Caffe, Torch, PyTorch, MXNet and more others. [20]

So now we can find easily what we need to work with machine learning. Some models are basic and well know. I choose some of them to try to predict the next position of the users.

5.2.1 Linear

The linear model is clearly the most known model in statistics and machine learning. It's generally the first model that people begun to test and learn basic stuff in machine learning. So a linear model is basically a model which assumes a linear relationship between the input variables and the single output variables.

The output can be calculated from a linear combination of the input variables. So basically the representation mathematic for a simple model is $y = B0 + B1 \times x$. Where x represents the input, y the output, BO and $B1$ coefficients.

Sample of Linear Regression - Height vs Weight

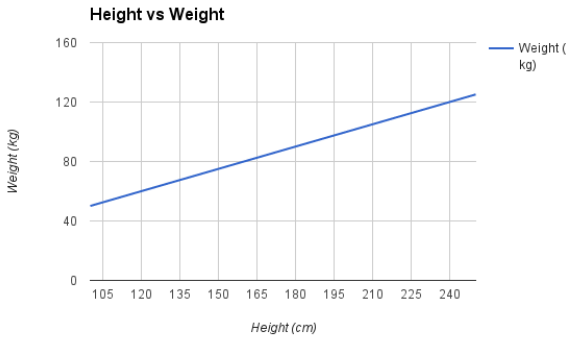

Figure 2: Source: www.machinelearningmastery.com

So using a linear regression model means estimating the values of the coefficients used in the equation with the data that we has available. We use usually the Ordinary Least Squares procedure to seek to minimize the sum of the squared residuals. This means that given a regression line through the data we calculate the distance from each data point to the regression line. Then we square distances, and sum all of the squared errors together. This is the quantity that ordinary least squares seek to minimize.

Distance to square of a Linear Regression

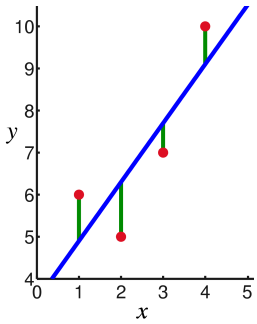

Figure 3: Source: wikipedia

So we optimize the values of the coefficients by iteratively minimizing the error of the model on our training data. This operation is called Gradient Descent and works by starting normally with random values for each coefficient. The sum of the squared errors are calculated for each pair of input and output values. We also use in this a learning rate as a scale factor and the coefficients are updated in the direction towards minimizing the error. The process is repeated until a minimum sum squared error seems to be obtained. So at this moment, we can say that the model has converged. *(See [3] for more details)*

Basically we use the model with the coefficient updated and test it on test data and train data to see how it works. And we compare training accuracy and test accuracy of the model.

5.2.2 Logisitic

The Logistic Regression model is similar to the previous Linear Regression Model. It is also one of the most famous models. It is based mainly on the logistic function, also called the sigmoid function $1/(1 + e^{-value})$. This function was named by *Pierre François Verhulst*, who firstly used it to represent the population growth, where the first stage of growth is exponential, then the growth slows, and finally stops.

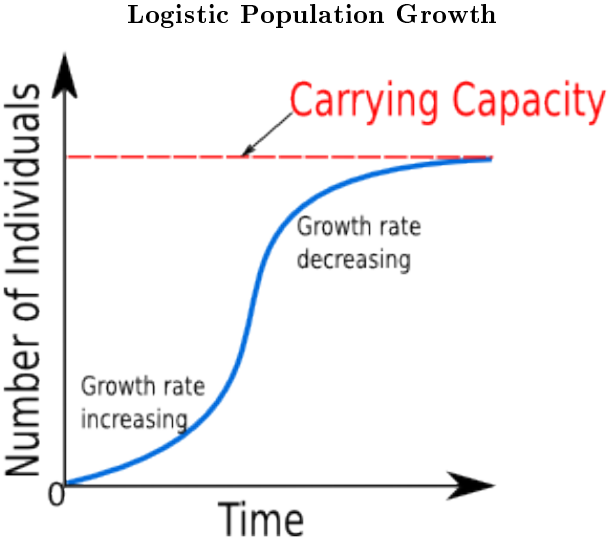

Figure 4: Source: www.study.com

So basically the function in our case can be represented as this $y = e^{(b0+b1*x)}/(1 + e^{(b0+b1*x)})$. Where y is the predicted output, $b0$ is the bias and $b1$ is the coefficient for the input value. So each column of our input data has a coefficient that we need to determine with the train step as I explained before for the linear model. So we try to optimize cost function at each iteration and update the coefficients values. *(See [4] for more details)*

5.2.3 Neural Network

The Neural network model is quiet different. It's a more complex model and really interesting one. The basic idea behind a Neural Network (NN) is to simulate interconnected brain cells inside a computer. So we can get the machine to learn things, recognize patterns, and make decisions in a humanlike way. The amazing thing about a neural network is that you don't have to program it to learn explicitly: it can learn by itself, like a brain !

The different cells that simulate neurones are classified in 3 different classes: input units (information from the outside world that the network will attempt to learn about), output layer (on the opposite side of the network, is how the NN responds to the information it's learned) and in the middle is a or some hidden layer(s) (which form(s) the majority of the artificial brain).

Sample of a Neural Network

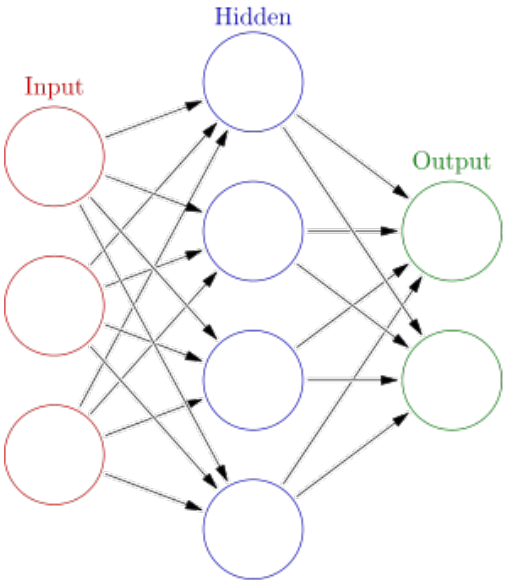

Figure 5: Source: wikipedia

The connections between one cell and another are represented by a number called a weight. Weights represent the influence that one unit has on another. So more the weights are high more the selected cell influence the cell linked.

A Neural Network with weigths

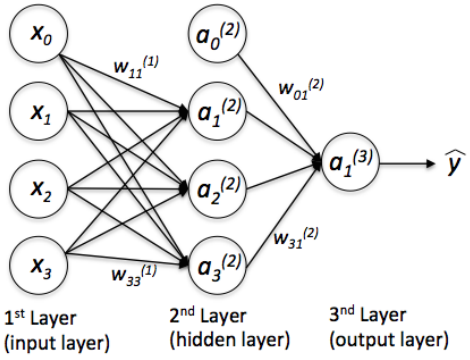

Figure 6: Source: www.kdnuggets.com

So how the NN learns really ? First it simply uses the basic method called feedforward. So each unit receives inputs from the units to its left, and the inputs are multiplied by the weights of the connections in which they go through. Every cell adds all the inputs it receives, computes some score and triggers the units it's connected to in order to give it's results.

Secondly, the neural network needs to do backpropagation, so a feedback process, in order to compare the output produced with the real output: The NN uses the difference between them to modify the weights of the connections between the cells in the NN in order to be nearest the good results the next time. *(See*

[21] for more details)

A Neural Network with feed forward and back propagation

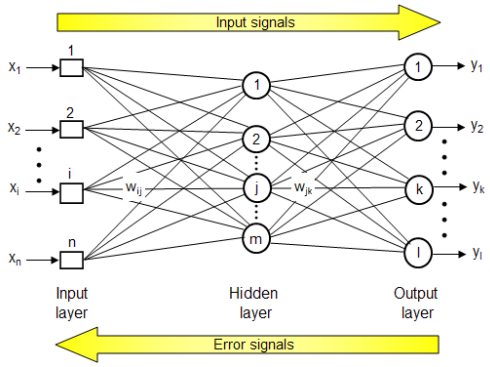

Figure 7: Source: www.researchgate.net

5.2.4 Recurrent Neural Network

The model Recurrent Neural Network (RNN) works like a simple NN. RNN is also similar while learning, except that it can learn also when we need to have persistence. Indeed when we need to work with previous events, previous data to predict the next one, it's where RNN becomes really usefull. We can see a RNN as multiple copies of the same Neural Network, each passing a message to his successor. So we have a persistence of some past events. *(See [13] for more details)*

A Neural Network with feed forward and back propagation

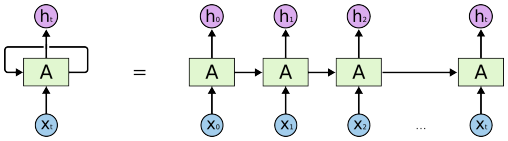

Figure 8: Source: colah.github.io

5.2.5 Long short-term memory

The Long Short-Term Memory (LSTM) is a special kind of RNN, which is capable of learning long-term dependencies. The LSTM does have the ability to remove, select and add pertinent information from past events. It's the main model of RNN used in practice.

5.3 Results

So now let's speak about the results of model used. The model was set up to try to predict the next positions of a user based on his last postion or some of his last postions. First I set the same simple parameters for all models used. So I set learning rate, bach size, number of epochs, training size, input, output, timesteps, hidden layers and neurons. In my case the parameters are defined and fisrt set as following:

Learning rate = 0.0001 : It can be represented as the speed that the model tries to correct it's prediction between training cycles. On one hand, higher the learning rate is, faster the model will change and adapte his prediction, but it will also may be really put the model in error by going too fast. On the other hand, smaller is the learning rate, slower the model will converge near its best value, but the model will be less likely to miss his apprenticeship and be in error.

Batch size = 20 : It corresponds to the number of input data given to the machine in the same time during taining the sessions. So if we set it to 20 and we have 100 input data of training, we will have 5 small cycles of training (one for each batch) in order to complet one entire cycle of training (named epoch).

Number of epochs = 1000 : One epoch represents a complete cycle of training. So when we have done one epoch, we have given all training data to the machine to learn. So indeed the number of epochs corresponds to the number of time all training data will pass trough the training.

Training size = 0.7 : It is the percentage p of all our data that will be used for training. So $1 - p$ will be the percentage of data that will be used for testing.

Input : It's a vector of 238 binaries which represents all the index of POIs of users. So for one current position only one of these binaries is equale to 1, which corresponds to the index of the position.

Output : It's also a vector but instead of 238 binaries, it is of 238 probabilities. So it corresponds to the probability of being in one each of this 238 POIs in next position. Obviously it is only the prediction of the model. But we can then take the maximum probability obtained from these different POIs and say that the model predicts this one as the next position.

Timesteps 1 until 5 : It corresponds to the number of previous positions that will be taken in account to predict the next position. So it represents the history of the user movements that will be used to predict his next position. For instance, a timesteps set to 2 corresponds in my model to predict the next positions based on the two last positions.

Hidden layers = 1 : It is the number of hidden layers, so it can represent the depth of the brain, its size or it's complexity.

Neurons = 20 : It is the number of neurons per hidden layers. Where each neuron tries to compute score based on the weights of theirs relations between them and other cells. So more neurons mean more things trying to be captured by the model on the training data (but doesn't mean that it will capture useful things).

5.3.1 Linear

The Linear model was realy a basic model. So I didn't expect it to work really well. The first result confirmed my thoughts. We can see that the movement predicted was clearly not good. They didn't really fit with the true trajectories linked.

Linear model: True movement vs Prediction

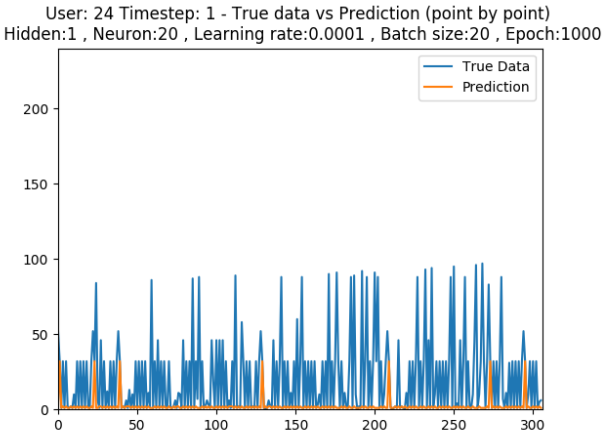

Figure 9: Generate with matplotlib

So then I tried with more timesteps until 5. It seems to preform really better, even if I used a linear model.

Linear: 1 timestep vs 5 timesteps

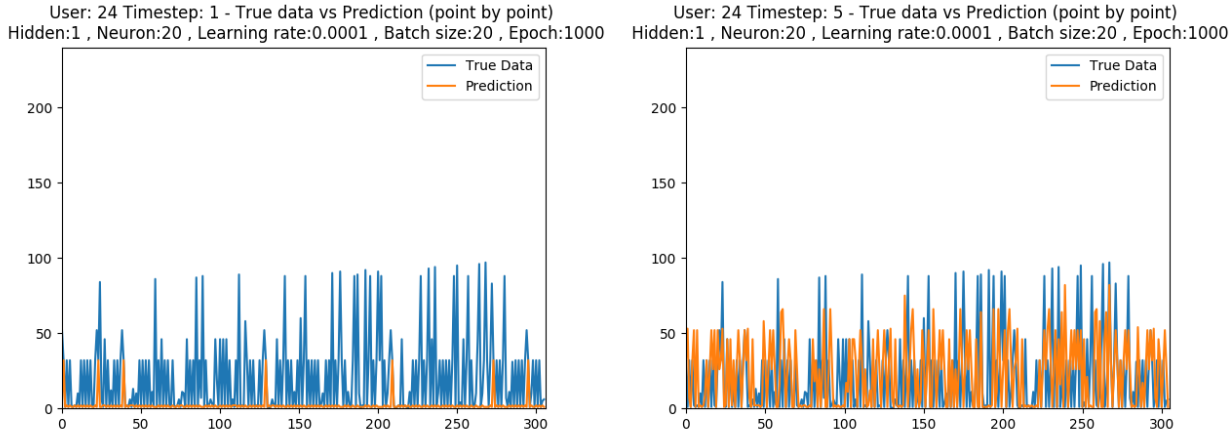

Figure 10: Generate with matplotlib

These first graphs and results seem finally not so bad, but when if we want to perform better and improve this model, it will be very hard. Indeed this model is a very simple one, without much parameters to change or adapt. So we can conclude that this model allow to show us that even now we can see that some timseteps can really improve results, but we can't continue to use it for capturing the behaviour of a humain movement. We need a model that can capture more informations.

5.3.2 Logisitic

The Logistic model was also quiet a basic model. So as the linear model I didn't expect much of it and the results also proved it to me.

Logistic: 1 timestep vs 5 timesteps

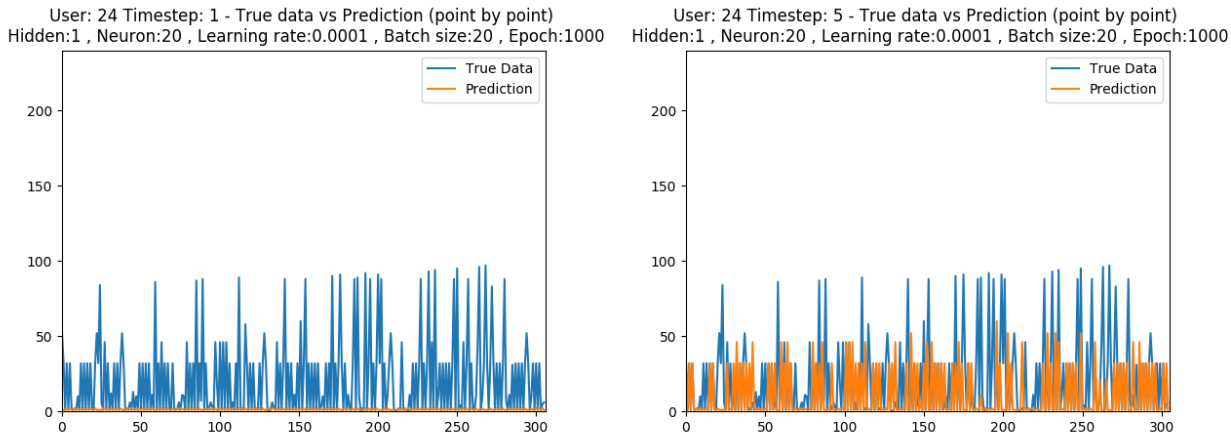

Figure 11: Generate with matplotlib

Same problem as the linear model, it show us that having more timesteps is already better, but really work with will be hard. Its results seem too much mechanical and without the human touch and will be hard to really improve it, because of its lake of parameters and flexibility. So it conclud also the need of a model more complexe and that we can really optimize.

5.3.3 Neural Network

First with the neural network, I was so mad and upset of my first results.

image

It seems to be less accurate than the basic models (linear and logisitc) for prediction. But aftera second look to really check what was predicted and what was the real output, I saw that the prediction seemed realy more near a human behaviour.

NN: 1 timestep vs 5 timesteps

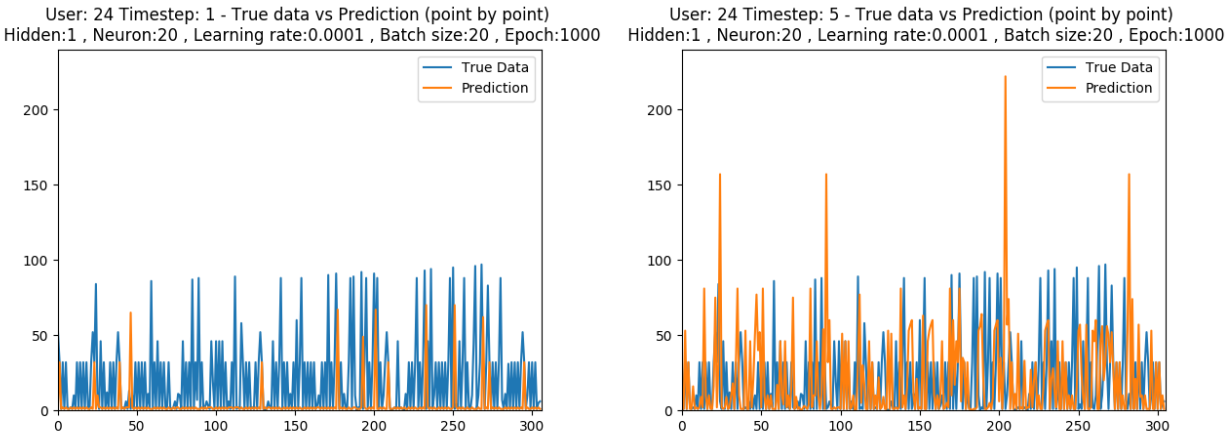

Figure 12: Generate with matplotlib

We can see that even with some basic settings the model is already trying to catch a more real behavior of movement than a simple model as the Linear and the Logisitc.

5.3.4 Long short-term memory

For the LSTM model, nearly the same effect occurs when we look at trajectory predicted by the model.

LSTM: 1 timestep vs 5 timesteps

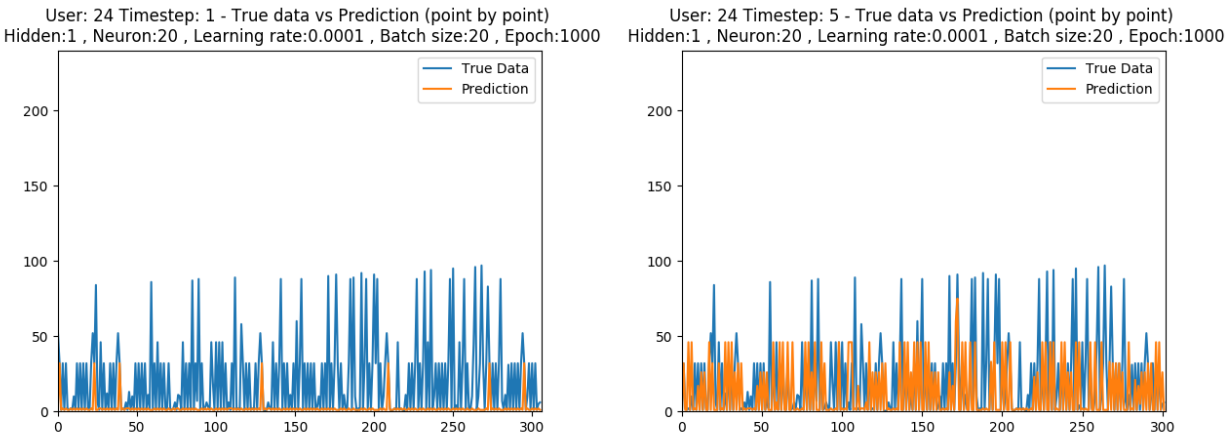

Figure 13: Generate with matplotlib

After having seen the real behaviour and compared it with one of the basic models, we can only confort our first tougth; so a more complex model will help us to have better results. Indeed wwe will be able to inprobve the model, because it has know only very simple parameters (1 hidden layers and 20 neurons only).

Even more in the next phases, we tried to show that having some memory of past information can really help us to have even better results, even we saw it already a little bit.

5.4 Discussion

Indeed in all my models, we can see that they capture some informations and behaviour from the data given to them, but only the Neural Network and the Recurent Neural Network seems to give us something near the reality more often than basic models. We can always perform better by trying to change some of the parameters and find maybe a better way for all these models before choosing one, but already with some tricky parameters we can see that results lean in favor of a more complex model than the basic ones. Moreover on the basic models we have less parameter to really improve results. Indeed with parameters of the Logisitic model and Linear model we can only try to change some basic stuff for their training as the number of epochs, the learning rates, so without really improve or really change them. It would as tring to improve a model that is already near its full capacity, so very complicate. In contrary of the NN and LSTM which can really be modify by changing their complexities by changing their number of neurons or even their number of hidden layers, so basicly their size and power. So it's why, I choose not to spend more time with a basic model and try next to optimize the Recurent Neural Network: my LSTM.

Why not also the NN that seems also really promising ? It is just that what I done with the basics models and the NN was just tring to simulate the potentiel of the LSTM when I used some timesteps. So tring to continue with the NN will juste to simulate the work of the LSTM. It's not really usefull, and even more the way I give it data for simulate some timesteps was only time consuming during training sessions. Indeed the NN and the two basic models, having input really larger when working with more timesteps. They have an input of 238 binaries multiply by the number of timesteps, where each 238 binaries represent one of the last positions. So with this practice I simulate timesteps of the LSTM and also have inputs clearly larger. So the time that these models take to proces it can also increase. So it's also why I choose to clearly continue with only the LSTM.

6 Optimize

The next big step was to try to find the best or at least good parameters for my models, parameters that can give me interesting results. So set and test some parameters then try to change them. For this, I tried to select some different range of values for my parameters and try to compare their results.

For instance I choose to set some parameters as 5 hidden layers, save the results, and then try to increase this number of layers. If the new result was better than the last one, in the next step I continued to increase it, but on the contrary if the results were worse, I decreased the parameters. And then I followed the same process for some of the other parameters.

It was clear that this task could take a lot of time, if we know that there are 150 data's users to train, so as much models to train. Having time and computer power limited, I choose to only train and make results for a part of my data. I choose to take only 25 users, but which ones ? Randomly ? Try to choose only users with different behavior in order to capture as much different types to take my decision for parameters ? But it would need a lot of time to define what would be the criteria for classifying the type of users.

So I choose to assume that the users with more transition between POIs would be better than those with

few transitions. Why ? One fact was that some users has really few data, so few transition between POIs, that using a big model of machine learning on them as LSTM would be absurde knowing that this type of machine learning needs normaly a lot of data to be correctly trained. The second fact was that users with more movements would have also normaly more transitions between different POIs. So it would represent globaly better my users and their areas visited.

Thereafter I obtained also different results for each of my 25 users selected. And I needed to find a way to say if the model performed good or bad or at least find out if with these parameters it performed better than with other parameters. So I choose two simple ways to evalute the results:

1. Manually by simply looking at the movements predicted and comparing them with some real one.
2. Statistically by selecting some mesures that seemed interesting. So these measures were used on the movement predicted and the real one, then compared with each other. Some of these measures have been then again compared with those of results based on other parameters, and I choose the parameter that seemed to have measures nearest to those from true data.

So this two processes are not really purely scientific and are also based on my own opinion, but it was easier and a fast way to choose between parameters.

6.1 Results

Concerning my parameters in comparison with the ones in the section *Prediction*, I just increase the batch size and the number of epochs, because now I will train the machine with only user that have more transition, so more data. Moreover I will use more complex model with higher parameters, so it will need more training cycles to caputure the behavior. Furthermore having more training cycles (epochs) will drastically increase the time spent on trainings, so I choosing to increase batch size that will speed up the process wihtout having really negative effects on the results. It will be maybe even be better with a higher batch size. Indeed according to some forum and discussion read higher the batch is, better is the accuracy of the estimate of the gradient. Having only a batch size of 20 as before was clearly only to process on user with really few transition, but now I restricted my job with only users with more transitions. So the batch size can be higher easily. So I set the other parmeters:

- Learning rate = 0.0001
- Batch size = 100
- Number of epochs = 5000
- Training size = 0.7
- Hidden layer = 1
- Timesteps = 5
- Neurons = 20

Then as I said some measures take in consideration to comare the results of my different settings. These measures were extract from the predict path from my models. So the path constits of each points predicted by the model based on the last real position or some of the last real positions. Moreover most of this features were extract with *Cesium* which is a library specialized for time-series.

Mean: Mean of observed values.

Median: Median of observed values.

Maximum: Maximum observed value.

Minimum: Minimum observed value.

Unique: Total number of unique observed values.

Std: Standard deviation of observed values.

Skew: Skewness of a dataset. Approximately 0 for Gaussian data.

Amplitude: Half the difference between the maximum and minimum magnitude.

Max slope: Compute the largest rate of change in the observed data.

Correlation: The Pearson correlation coefficient measures the linear relationship between two datasets.

P-value: The p-value roughly indicates the probability of an uncorrelated system producing datasets that have a Pearson correlation at least as extreme as the one computed from these datasets.

6.1.1 Hidden layer(s)

First, I tried to find the best number of hidden layers so I tried different numbers: 1,3,5,10,20 and 40. The result was pretty obvious that between 5 hidden layers were clearly better than less or more layers. We can see some features extract from generate path predicted (point by point) of my different results and the the real mean based on true movement from my users in the following table. We see clearly that 5 hidden layers perform a little bit better than the others. Especially when we look to the *Median* which is with *Correlation* the measures that I look in prority.

Some measures from different number of hidden layers and true movement

	1 HL	3 HL	5 HL	10 HL	20 HL	40 HL	Real
Mean	13.94	15.083	14.435	8.66	7.199	1.32	22.012
Median	4.64	9.54	7.88	2.88	8.04	1.32	9.52
Maximum	115.0	105.32	103.56	87.68	52.12	1.32	133.12
Minimum	0.2	0.2	0.24	0.28	0.36	1.32	0.12
Unique	27.44	27.88	21.44	12.0	5.12	1.0	47.08
Std	21.194	21.837	21.742	15.181	7.447	0.0	33.279
Skew	3.247	2.837	3.198	5.054	5.627	0.0	2.226
Amplitude	57.4	52.56	51.66	43.7	25.88	0.0	66.5
Max slope	112.64	103.04	100.2	86.68	51.4	0.0	130.48
Correlation	0.098	0.039	0.039	0.031	0.005	-1.0	1.0
P-value	0.206	0.496	0.454	0.374	0.494	-1.0	0.0

Figure 14: Source: Generate with matplotlib

Then for instance we can see that with the following graphs that compare movement generated point by points that 5 hidden layers is better than a unique hidden layer. We can see that the model with 5 hidden layers try to capture more transitions and clearly more differents positions.

LSTM: 5 Hidden vs 1 Hidden layers

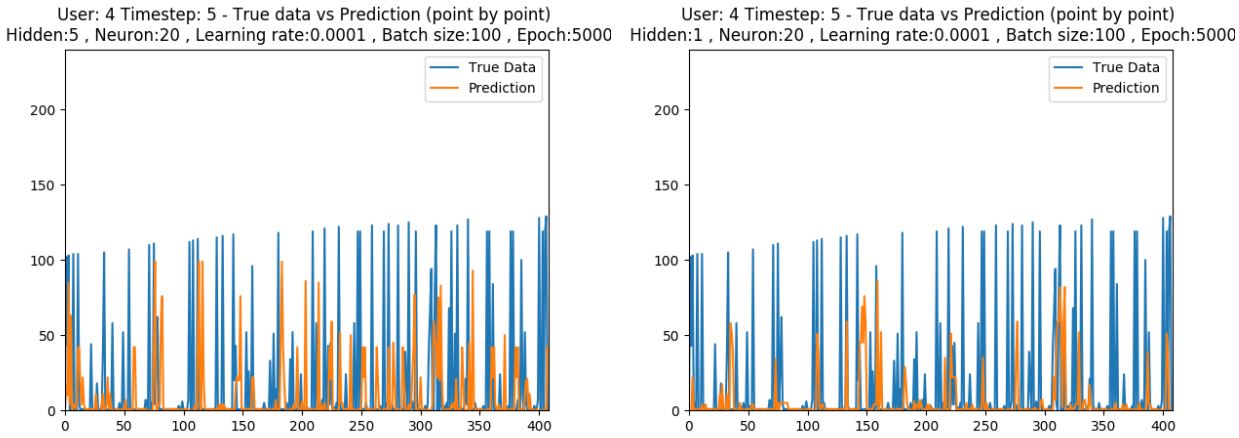

Figure 15: Generate with matplotlib

Then we can also see that more hidden layers deacrese a lot the results. It's seem that the model with too much hidden layer begin to only predict the POIs with more occurences. We can presume different effects. Maybe there is too few training cycles or even that the model is too complex for this data as we try to caluclate a simple addition with a powerfull computer while a simple calculator can do the job.

LSTM: 5 Hidden vs 20 Hidden layers

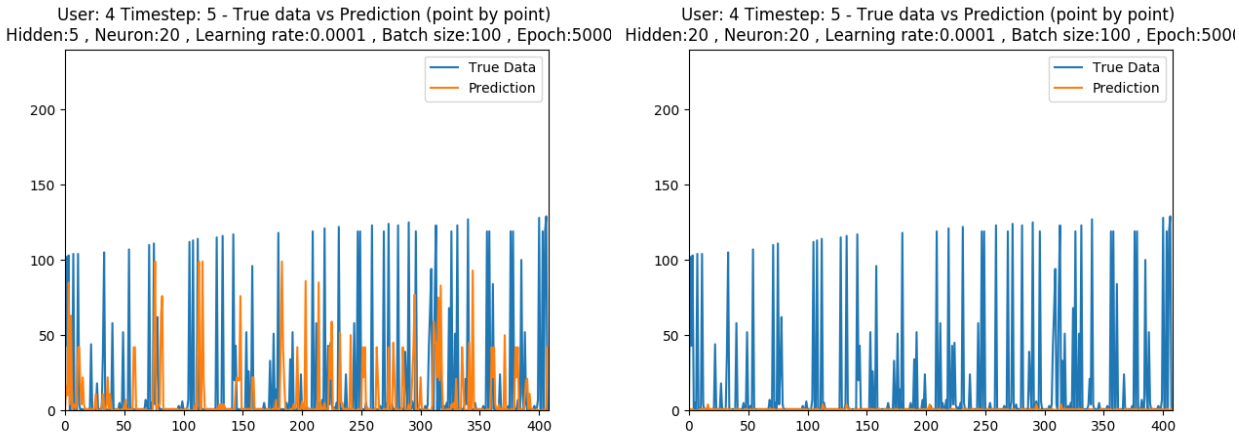

Figure 16: Generate with matplotlib

However all these results seem to be in favor for 3 or 5 hidden layers, but accordingly to some predictions of some users, it seems better to keep 5 hidden layers. So I keep this parameters for the following.

6.1.2 Neurons per hidden layers

So then I tried to find a good number of neurons per layer. It was set to 20 during the last tests, so then I tried to increase, decrease and compare them: 10,20,50,70 and 100 neurons per layers. So I found that the best number of neurons was approximately 50.

Some measures from different number of neurons and true movement

	10 N	20 N	50 N	70 N	100 N	150	Real
Mean	10.151	14.435	17.843	17.417	17.655	17.398	22.012
Median	5.28	7.88	8.48	4.92	5.86	8.3	9.52
Maximum	91.08	103.56	110.2	110.28	111.6	121.72	133.12
Minimum	0.28	0.24	0.16	0.16	0.2	0.16	0.12
Unique	11.08	21.44	36.2	38.28	36.0	36.6	47.08
Std	15.604	21.742	25.169	25.22	25.492	25.142	33.279
Skew	5.071	3.198	2.489	2.378	2.456	3.175	2.226
Amplitude	45.4	51.66	55.02	55.06	55.7	60.78	66.5
Max slope	88.84	100.2	107.6	106.2	109.96	119.44	130.48
Correlation	0.026	0.039	0.056	0.047	0.041	0.031	1.0
P-value	0.448	0.454	0.315	0.46	0.45	0.417	0.0

Figure 17: Source: Generate with matplotlib

We can see it here that to few neurons per hidden layer will clearly decrease results.

LSTM: 50 neurons vs 10 neurons

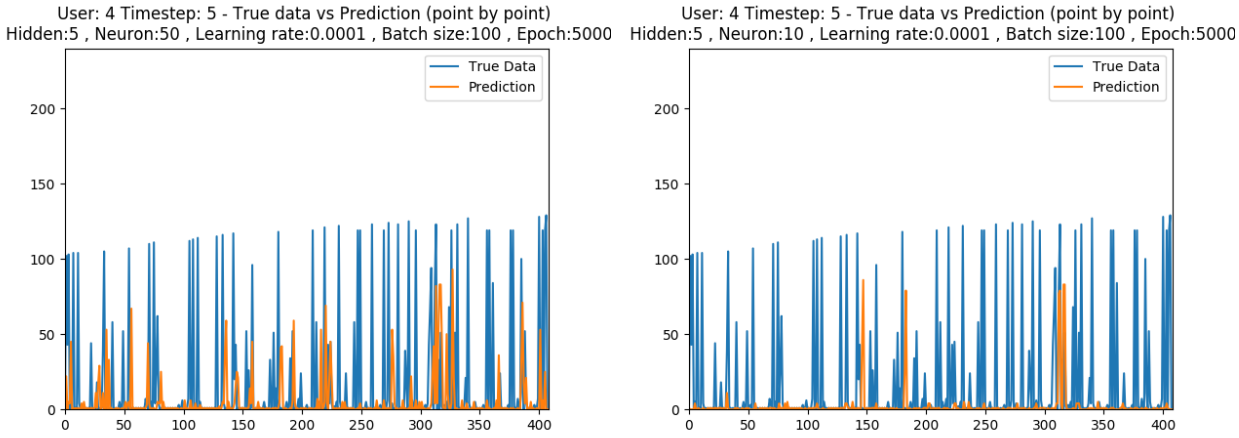

Figure 18: Generate with matplotlib

In the same way to much neurons will not really improve results and will just increase drastically the training time.

LSTM: 50 neurons vs 100 neurons

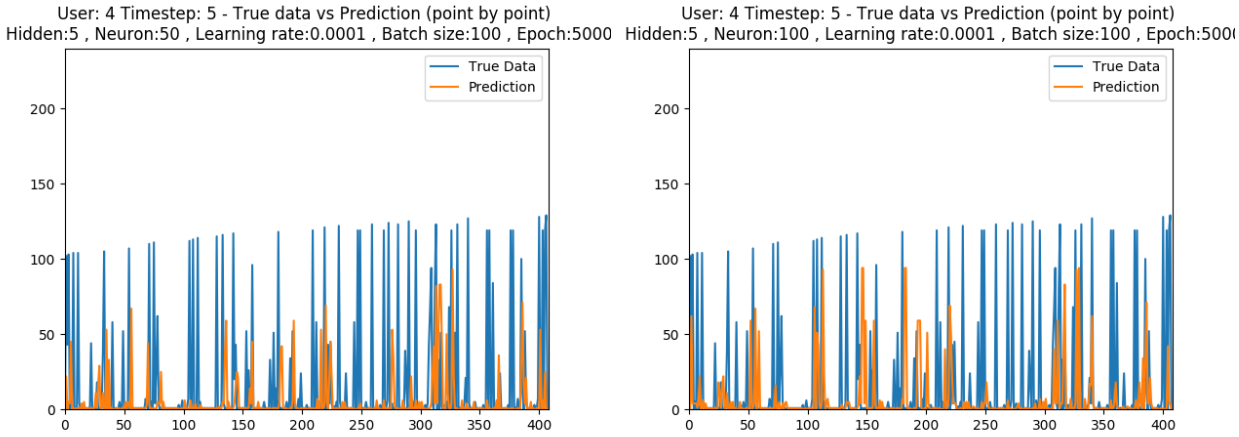

Figure 19: Generate with matplotlib

So I set neurons per hidden layers to 50 for the following.

6.1.3 Timesteps

Then I tried to find the best timesteps, so the best memory of last positions to take in account for predictions. So I tried different timesteps with my previous best parmeters: 1,3,5,7 and 10 timesteps. It resulted that 5 timesteps is better with our mesures.

LSTM: 5 timestep vs 1 timesteps

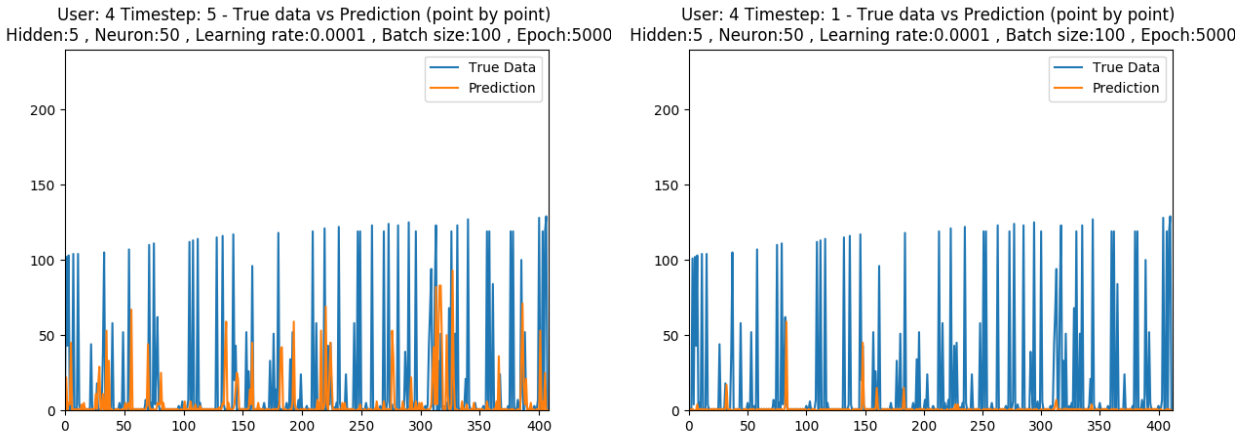

Figure 20: Generate with matplotlib

It is less obvious that too much timesteps is bad with the following graph.

LSTM: 5 timestep vs 10 timesteps

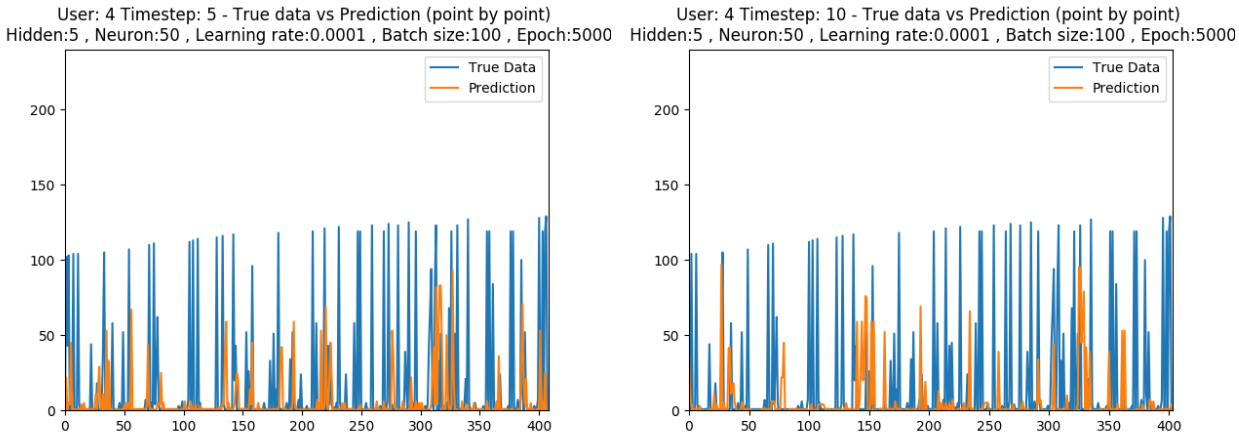

Figure 21: Generate with matplotlib

Indeed for this paricular user the model with more timesteps seems good, but when we look at our measures we see clearly the superiority of having 5 timesteps.

Some measures from different number of timesteps and true movement

	1 T	3 T	5 T	7 T	10 T	Real
Mean	4.068	13.565	17.843	17.47	17.692	22.057
Median	1.96	3.38	8.48	5.64	5.36	9.52
Maximum	62.2	107.12	110.2	108.48	110.68	133.12
Minimum	0.28	0.16	0.16	0.16	0.2	0.12
Unique	6.44	27.72	36.2	38.36	36.32	46.64
Std	7.008	22.41	25.169	25.47	25.896	33.329
Skew	6.444	3.097	2.489	2.318	2.296	2.229
Amplitude	30.96	53.48	55.02	54.16	55.24	66.5
Max slope	60.68	105.16	107.6	106.88	108.2	130.48
Correlation	0.034	0.079	0.056	0.01	0.017	1.0
P-value	0.342	0.282	0.315	0.565	0.494	0.0

Figure 22: Source: Generate with matplotlib

6.2 Discussion

All these results show us that it can be realy tricky to find the best parameters. Moreover that trying all combinations of parameters can't be done without a lot time spent on training on the machine. So we need to simplify the problem.

We also saw that some parameters obtain better results on some measures than my final selected parameters, but on average worst. Indeed machine learning as RNN can be really complicate and the results are sometimes surprising. It's really hard to know exactly why some parameters work better on some measure than other parameters. With my test, we can still conclude some points:

1. Some hidden layers can capture better long terme dependencies
2. Too much hidden layers will clearly increase a lot the time spent for training without really helping to capture behavior
3. Not enougth neurons will result to having really similar results than a simple model as linear or logistic. Indeed it will be as having a model which could capture only too few informations on the data.
4. To much neurons will, as hidden layer, increase a lot the time spent on training and have worst results.
5. Only one timestep will never capture long term dependencies. It will result in a simple markov chain during the prediction, so with only the current state (position) that will influence the prediction. So it will not, in our case, try to understand the movement of users, but only predict the most likely next postion based on occurences.
6. Some timesteps will really try to capture and understand the movement of users. It will know some previous postions to really predict the next postion based on the previous path.
7. To much timesteps will not really improve the results. Moreover it will maybe overfit a specific trajectory and predict a linked specific position. While the trajectory needs may be to go out of this specific trajectory.

7 Generate

Finally to generate the trajectories, I used my training model with the parameters that seemed better to me on the previous section. So prediction based on a LSTM model with parameters set to:

- Learning rate = 0.0001
- Batch size = 100
- Number of epochs = 5000
- Training size = 0.7
- Timesteps = 5
- Neurons = 50
- Hidden layers = 5

So my model can give us a vector of 238 probabilities for the next position based on the last 5 positions (timesteps 5). The sum of all 238 probabilites equal 1. So I have two ways to generate the trajectories:

1. By taking every time the last 5 positions and select the position that has the highest probability in our output vector of 238 probabilities.

2. By taking every time the last 5 position and select randomly between the 238 positions based on their probabilities in the output (the vector of 238). So a position with a higer probability will have more chance to be generate for the next position than a position with a low probability.

The first method is presumed better when we try to generate few steps. Indeed it will generate the most likely next positions in term of probability, but it is also more susceptible to go in a wrong cycle by predicting always the same movement.

The second method can be better to go out of the same cycle of movements, because even some POIs that are uncommon will be sometimes (rarely) predicted, so genereted. This behavior seems more logical if we want to predict large steps, because a user's behavior goes sometimes out of the box, so out of his recurrent path.

7.1 Results

The results compare different paths whether real or generated in some way and compare them.

7.1.1 Random Generator

So firstly, I compare a true path of a user and a path generated randomly.

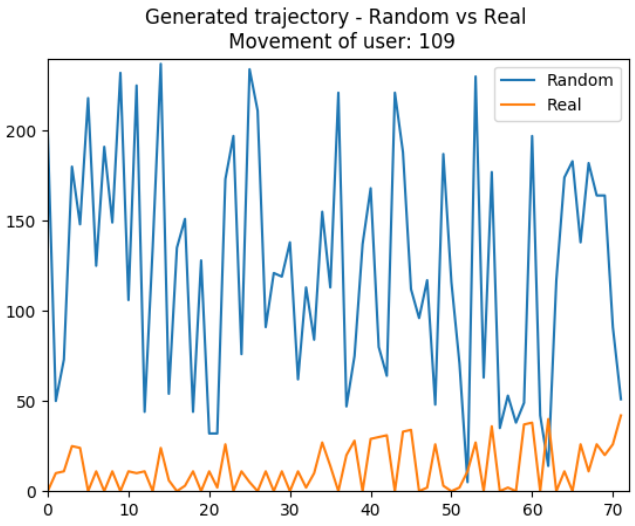

Figure 23: Source: Generate with matplotlib

We can clearly see that, as we speak about the Random Walk Model, a generator based principaly on random can not capture a real or similar behavior of a human. It's really obvious that for all spoken applications of trajectories, we can really not take a generator as the one above.

7.1.2 Best Probability Generator

So now we compare a real path with a generated one based on my LSTM model, which gives us probabilities for the next positions. We simply always take the POIs with the highest probabilites.

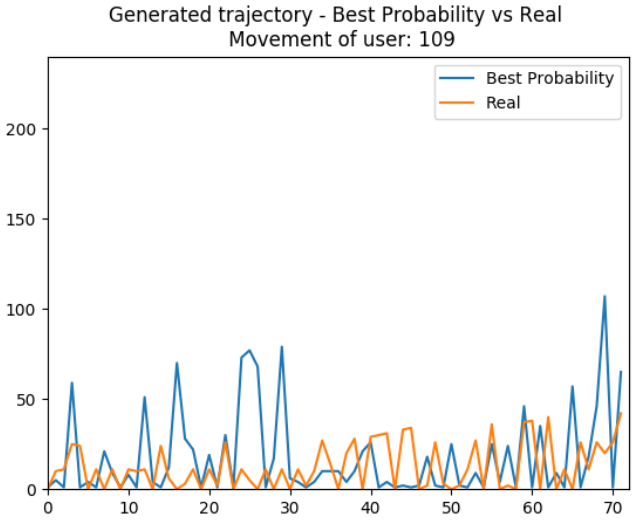

Figure 24: Source: Generate with matplotlib

We see that this generator is really better than the random one. We can see that he tries to capture a human behavior, but if we increase the number of step to generate.

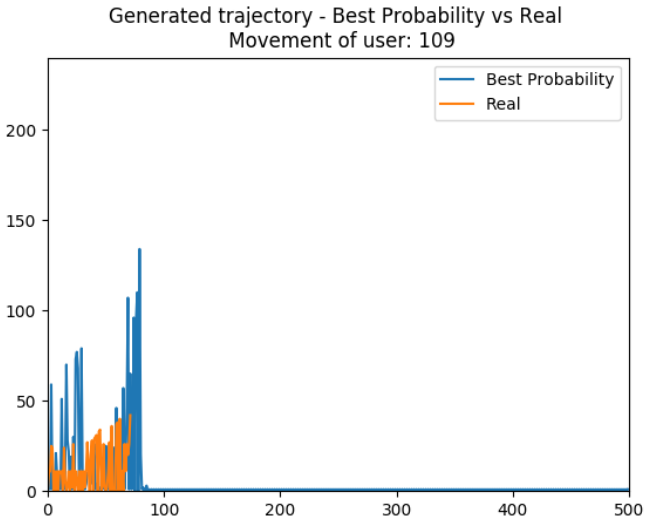

Figure 25: Source: Generate with matplotlib

We can see that the generator can fall into a repeated cycle of the same movements or even worst only a unique position.

7.1.3 Random Choice Probability Generator

The results of this generator will also be compared to a real trajectory. This generator is based on the LSTM model, but instead of just taking the best probabilities it is also based a little bit on random. It combines probability and random choice. So it will use the LSTM model to have the probabilites of all POIs for the next positions and it will take as next position randomley between this POIs based on their probabilites. So a POIs with a higher probability will more likely be chosen.

So why use random while we have seen that random can not capture a real behavior ? Because the method allows the path generated not to go in a repeated cycle of the same movements. Moreover is not totally random because is based on the probabilites given by the LSTM model. So it's just a way to try to simulate the trajectories to go sometimes in another way than the most recurrent.

We can think of a man going every day to work; he will nearly every days go by the same path, but may be sometimes will choose or be forced to take another path. May be because of too much traffic jam, may be because needs to look for somebody or even just simply because he is bored by this same path or because he want try a new path. Even if, as I said, this generator uses random, we see also that it is firstly based on the LSTM predictions, so it uses random based on behavior captures from users.

The generator give us this following results.

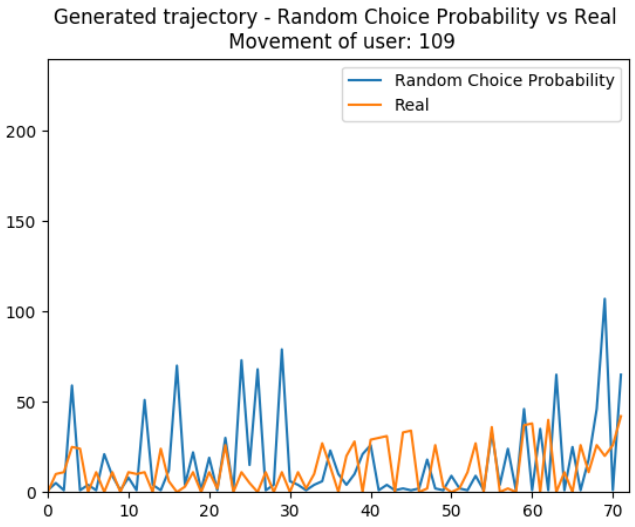

Figure 26: Source: Generate with matplotlib

It is really better than the generator random, and seems near the generator based on only best probabailities.

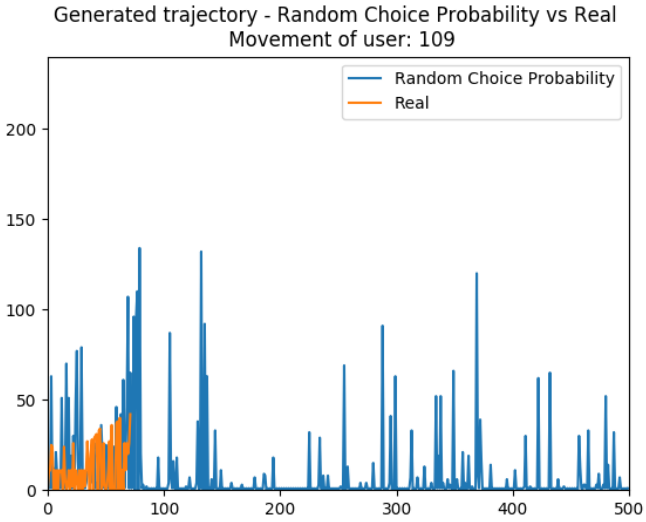

Figure 27: Source: Generate with matplotlib

In contrary than the lastgenerator, we can also see that it performs really better when we try to generate more steps. Indeed this generator will fall less in a restricted same cycle of movements or unique positions.

7.2 Discussion

With these results we can cleary consolidate the first tought that a generator based only on random is really not recomanded in synthectic trajectories. Indeed only bad results will be obtained in the applications mentionned before as traffic management.

All my results also consolidate my tought that some memory of past positions can help for better generate capture behavior of people. Indeed the results on generators based on the LSTM model shows us that we clearly obtain better results than random or just taking best occurrences.

It shows also that taking only purely the higher probability given by the LSTM model for the next position is not perfect, particuliary when we try to generate large steps. It can give us the problem of fall in a pariculiar cycle without a lot of vairances, so a monotone behaviour without the little magic of the human kind for change.

The results conclud also that combine just a little bit of random in the generator can really help to produce a more human-like path.

It's really hard to say which one between the two generators has to be used: based only on best probablity or with some random. It depends on what trajectories we need and on what kind of purpose we need it. For instance if we need to predict only a little projection of the movement of one user, it will be better to use only the Best Probability Generator. Conversely if we need to may be simulate a large number of human movements, for example in traffic simulation, it will be better to use the Random Choice Probability Generator.

8 Conclusion

In conclusion we can clearly approuve the common thought that machine learning can be usefull in many areas. In our case it can clearly help us for trying capturing real behavior of users and trying to summerize them. The present paper shows us that in some points the machine can capture and anderstand some human behaviour in a context of geospatial trajectories. Even if the present work uses only simplification of the problem by using only sequences of areas, we can see the potentiel of machine learning, particulary the power of LSTM. It will be really interesting to go further in the context of geospatial trajectories and try to perform better by adding and testing the impact of more features; time, delta time, longitude, latitude, and so one. My work for the moment ends on a global conclusion that we can now really use computer to analyse, predict, cluster and even capture human behavior. I hope that me or other people will go further and apply a better model on the subject in order to finally create a generator that will be really usefull on the fields of synthetic geospatial trajectories.

9 References

[1] Jason BROWNLEE. "Gentle introduction to models for sequence prediction with recurrent neural networks". Adresse : https://machinelearningmastery.com/models-sequence-prediction-recurrent-neural-networks/. consulted 28.12.2017.

[2] Jason BROWNLEE. "Gentle introduction to the adam optimization algorithm for deep learning". Adresse : https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/. consulted 28.12.2017.

[3] Jason BROWNLEE. "Linear regression for machine learning". Adresse : https://machinelearningmastery.com/linear-regression-for-machine-learning/. consulted 28.12.2017.

[4] Jason BROWNLEE. "Logistic regression for machine learning". Adresse : https://machinelearningmastery.com/logistic-regression-for-machine-learning/. consulted 28.12.2017.

[5] Jason BROWNLEE. "Mini-course on long short-term memory recurrent neural networks with keras". Adresse : https://machinelearningmastery.com/long-short-term-memory-recurrent-neural-networks-mini-course/. consulted 28.12.2017.

[6] Bertil. CHAPUIS, Arielle MORO, Vaibhav KULKARNI, and Benoît GARBINATO. "Capturing complex behaviour for predicting distant future trajectories". page 10, 2016.

[7] Kirill FUCHS. "Machine learning: Classification models". Adresse : https://medium.com/fuzz/machine-learning-classification-models-3040f71e2529. consulted 28.12.2017.

[8] João Bártolo. GOMES, Clifton PHUA, and Shonali KRSHNASWAMY. "Where will you go? mobile data mining for next place prediction". page 12, 2013.

[9] Danijar HAFNER. "Introduction to recurrent networks in tensorflow". Adresse : https://danijar.com/introduction-to-recurrent-networks-in-tensorflow/. consulted 28.12.2017.

[10] Erik HALLSTRÖM. "How to build a recurrent neural network in tensorflow". Adresse : https://medium.com/@erikhallstrm/hello-world-rnn-83cd7105b767. consulted 28.12.2017.

[11] Trevor HASTIE, Robert TIBSHIRANI, and Jerome FRIEDMAN. *The Elements of Statistical Learning*, volume 2. Springer, 2001.

[12] G. E. HINTON, N. SRIVASTAVA, A. KRIZHEVSKY, I. SUTSKEVER, and R. R. SALAKHUTDINOV. "Improving neural networks by preventing co-adaptation of feature detectors". page 18, 2012.

[13] Andrej KARPATHY. "The unreasonable effectiveness of recurrent neural networks". Adresse : http://karpathy.github.io/2015/05/21/rnn-effectiveness//. consulted 28.12.2017.

[14] Vaibhav KULKARNI and Benoît GARBINATO. "Generating synthetic mobility traffic using rnns". page 4, 2017.

[15] Zachary C. Lipton, John Berkowitz, and Charles ELKAN. "A critical review of recurrent neural networks for sequence learning". page 38, 2015.

[16] Michael NIELSEN. "Improving the way neural networks learn". Adresse : http://neuralnetworksanddeeplearning.com/chap3.html. consulted 28.12.2017.

[17] Christopher OLAH. "Understanding lstm networks". Adresse : http://colah.github.io/posts/2015-08-Understanding-LSTMs/. consulted 28.12.2017.

[18] MONIK PAMECHA. "A noob's guide to implementing rnn-lstm using tensorflow". Adresse : http://monik.in/a-noobs-guide-to-implementing-rnn-lstm-using-tensorflow/. consulted 28.12.2017.

[19] Warren S. SARLE. "How many hidden units should i use?". Adresse : ftp://ftp.sas.com/pub/neural/FAQ3.html#A_hu. consulted 28.12.2017.

[20] Amey VARANGAONKAR. "Top 10 deep learning frameworks". Adresse : https://datahub.packtpub.com/deep-learning/top-10-deep-learning-frameworks/. consulted 28.12.2017.

[21] Chris WOODFORD. "Neural networks". Adresse : http://www.explainthatstuff.com/introduction-to-neural-networks.html. consulted 28.12.2017.

A Annexe