Implementation of ”Active Learning for Estimating Reachable Sets for Systems with Unknown Dynamics” paper on Python

by

Aghakhan Huseynli

Bachelor Thesis in Computer Science

Dr. Amr Alanwar Abdelhafez
Name and title of the supervisor

Date of Submission: January 31, 2023

Jacobs University — School of Engineering and Science

With my signature, I certify that this thesis has been written by me using only the indicates resources and materials. Where I have presented data and results, the data and results are complete, genuine, and have been obtained by me unless otherwise acknowledged; where my results derive from computer programs, these computer programs have been written by me unless otherwise acknowledged. I further confirm that this thesis has not been submitted, either in part or as a whole, for any other academic degree at this or another institution.

Signature Place, Date

Abstract

In this work, we present an implementation of the paper ”Active Learning for Estimating Reachable Sets for Systems with Unknown Dynamics” on Python. The main goal of this implementation is to estimate the reachable set of a nonlinear system with unknown dynamics using active learning. The system dynamics are modeled by a nonlinear function that describes the evolution of the system's state. The oracle, which provides information about the reachable set, is implemented using a finite-horizon constrained optimal control problem. This problem is solved using numerical optimization techniques and the resulting reachable set is visualized using a scatter plot. The active learning algorithm, called greedy active learning, is used to iteratively select the most informative samples from a set of unlabelled samples to be labeled and added to the reachable set estimate. The performance of the algorithm is demonstrated through simulations of a nonlinear system with unknown dynamics. The results show that the proposed method is able to efficiently estimate the reachable set of the system using a limited number of labeled samples.

Contents

1 Introduction

The study of control systems has experienced a surge in the development of innovative techniques for the analysis and control of dynamic systems in recent years. A crucial challenge in this area is to accurately estimate the reachable sets of a system, which describe all the states that the system can attain starting from a particular initial condition. This information is extremely valuable in understanding the behavior of the system over time, and can be used for several important purposes such as identifying potential safety and performance issues, designing control algorithms to ensure desired system behavior, and analyzing robustness.

However, traditional methods for reachable set estimation, such as numerical simulation, can be both time-consuming and computationally intensive, especially for high-dimensional systems and require a closed-form analytical model of the system dynamics. To tackle this issue, researchers have put forth various active learning-based approaches for reachable set estimation, which is a machine learning technique that enables a model to learn from a limited number of labeled data points instead of being trained on a large dataset. Also, in this paper, we query the oracle batch-wise from a pool of unlabeled samples: this is referred to as 'pool- based batch selection' [1] and is generally faster than single-instance selection methods By reducing the number of simulations required, active learning can significantly reduce computation time and improve the efficiency of reachable set estimation.

In this paper, we present the implementation of active learning for estimating reachable sets for systems in Python. The implementation involves training a model using a limited number of labeled data points to predict the reachable sets of the system from unlabelled set. We also conduct a comprehensive evaluation of the proposed approach, comparing its performance with traditional numerical simulation-based methods on a set of benchmark problems.

The implementation of active learning for estimating reachable sets in Python has numerous potential applications in control systems engineering, such as the design of control strategies, safety analysis, and robustness analysis. This paper serves as a valuable resource for researchers and practitioners who are interested in exploring the use of active learning for reachable set estimation in control systems.

2 Preliminaries

2.1 Dynamical Systems

A dynamical system is a mathematical model that describes the behavior of a system over time, taking into account the interplay between its internal states and external inputs. The study of dynamical systems is a branch of mathematics that has applications in a wide range of fields, including physics, engineering, biology, and economics.

A dynamical system is "a set of points and a set of transformations such that each transformation moves some points to others in a definite manner." [2] In other words, a dynamical system is defined by the way its state changes over time in response to inputs.

We can divide dynamical systems to two types:

1. Continious time variable *t* where:

$$\frac{dx}{dt} = \widehat{x} = X(x)$$

2. Discrete time variable *t* where:

$$X_{t+1} = f(x_t)$$

   Note that, although we focus on nonlinear discrete-time systems in this paper for our implementation, the proposed method can be applied directly to continuous-time systems and/or linear systems.

2.2 Active Learning

Active learning is a machine learning approach in which a model is trained on a limited set of labeled data points, with the ability to actively query the data labeling process to obtain additional information. This approach differs from traditional machine learning methods, where the model is trained on a large, fully labeled dataset. Active learning has become increasingly popular in recent years, as it has been shown to significantly reduce the amount of labeled data required to train a model while still achieving high accuracy.

The goal of active learning is to effectively balance the trade-off between the need for labeled data and the cost of labeling additional data. In traditional machine learning, a large labeled dataset is typically used to train the model. However, collecting and labeling a large amount of data can be both time-consuming and expensive. With active learning, the model can actively select the most informative data points to be labeled, reducing the overall cost of labeling data.

One of the key advantages of active learning is that it can significantly reduce the amount of labeled data required to train a model, making it a more cost-effective

and efficient approach. This is particularly useful in situations where the cost of labeling data is high or the process of labeling data is time-consuming. In these scenarios, active learning can reduce the amount of resources required to train a model.

2.3 Oracle

In the context of active learning, an oracle is a source of ground-truth information used to train a machine learning model. It acts as a reference point for the model, providing the correct label for a given input. In active learning, the model queries the oracle for the label of a small set of unlabeled data points, which it selects based on some criteria, and uses this information to improve its performance on future predictions. The oracle can be a human annotator, a pre-existing dataset, or some other source of labeled data. The role of the oracle is crucial in active learning as it provides the model with the information it needs to learn and generalize to new, unseen data.

For example, in a reachability analysis scenario, the oracle might provide the labels for the states that the system can attain from a particular initial condition, and the active learning algorithm would use these labeled instances to train a model for estimating the reachable sets of the system. [3]

2.4 Reachability analysis

Reachability analysis is a field of study in control systems and formal verification that focuses on determining the set of states that a dynamic system can attain from a given initial state. This information is critical for various purposes such as understanding the system's behavior over time, identifying safety and performance issues, designing control algorithms to ensure desired system behavior, and analyzing robustness.[4]

There are several approaches for performing reachability analysis, including:

1. Numerical simulation: This is a traditional approach where the system's equations are integrated over time to obtain the set of states that the system can attain from a given initial condition. Numerical simulation can be time-consuming and computationally intensive, especially for high-dimensional systems.
2. Analytical methods: This approach involves using mathematical methods such as linear programming and convex optimization to obtain an analytical representation of the reachable sets. Analytical methods are often limited in their applicability to systems with simple dynamics and can be difficult to use for non-linear and hybrid systems.
3. Hybrid systems methods: These methods are specifically designed for systems with both continuous and discrete dynamics and can handle the interaction between the two. Hybrid systems methods often use a combination

of numerical simulation and analytical methods to obtain an estimate of the reachable sets.[5]

4. Model checking: This is a verification technique that uses formal methods to rigorously analyze the behavior of a system. Model checking can provide a complete and accurate estimate of the reachable sets, but is often limited in its applicability to large and complex systems.
5. Machine learning-based methods: This approach uses machine learning algorithms to estimate the reachable sets of a system. Machine learning-based methods are often faster and more efficient than traditional numerical simulation-based methods and can be used for high-dimensional systems. Active learning is one of the popular machine learning-based approach for reachable set estimation which we use in paper.

2.4.1 Types of reachable sets

The study of reachable sets involves the analysis of the set of states that a dynamic system can attain over a certain period of time. The reachable sets are commonly classified into two main categories: forward reachable sets and backward reachable sets.

Forward Reachable Sets: Forward reachable sets describe the set of states that a system can attain starting from a given initial condition and evolving over time under the influence of the system's dynamics. The focus of forward reachable set analysis is on predicting the future behavior of the system given a particular initial state. The computation of forward reachable sets is often performed using numerical integration techniques such as Runge-Kutta methods or Monte Carlo simulation.

Backward Reachable Sets: Backward reachable sets describe the set of states that a system must have had in the past to reach a particular final condition at a given time. The focus of backward reachable set analysis is on identifying the source of a system's behavior given a particular final state. The computation of backward reachable sets is often performed using set-valued methods such as set-valued integration, set-valued Lie derivatives, or set-valued backpropagation algorithms.

In addition to these two main categories, there are also other variants of reachable sets that have been proposed, such as the concepts of controlled reachable sets, maximum reachable sets, and minimum reachable sets, to name a few. The choice of the appropriate type of reachable set depends on the specific problem at hand and the desired level of detail and accuracy in the analysis.

Due to the lack of knowledge about the dynamics of the system, the backwards reachable set would be a suitable solution for our problem. Here is a more in-depth definition of the backwards reachable set:

- *The T -step backward-reachable set* $R_t \subseteq X$ *of a compact set* $\Omega \subseteq X$ *is the set of all initial conditions* $x_0 \epsilon X$ *for which there exists a sequence of inputs such that* $u_t \epsilon U$ *such that* $x_{t+1} = f(x_t, u_t) \epsilon X$ *for* $t = 0, ..., T-1$ *and* $x_T \epsilon \Omega$ *where* $T \epsilon N$*, that is:*

$$R_t(\Omega) = \{x_0 \epsilon X : \exists u_t \epsilon U,$$
$$x_{t+1} = f(x_t, u_t) \epsilon X, x_T \epsilon \Omega\}$$

3 Active Learning with Infallible Oracle

In this part of the paper we assume that information about the reachable set is available through an oracle and it always provides right information about the sample being in reachable set.

3.1 Greedy Active Learning

Estimating reachable sets with a greedy active learning (AL) algorithm involves iteratively selecting the most informative samples to be labeled in order to improve the accuracy of the estimated reachable set. The algorithm starts with a labeled set L_0 and an unlabeled set S in the state space X. A non-trivial initial classifier is generated using L_0 and S, which is usually obtained through gridding or sampling methods.

In each iteration, a set of samples $L'k$ from S is selected based on the prior classifier, and only these samples are labeled by the oracle and added to the accumulated labeled set. The goal is to select the N_s most informative samples iteratively in batches. The informativeness of the samples in S is quantified using the Shannon entropy. The entropy is high for samples with high uncertainty and low for samples with low uncertainty.

$$J_E(S) = -\sum_x \sum_y p_k(y|x) log_2 |p_k(y|x)|$$

$$\text{given that } x \in S \text{ and } y \in \{-1, +1\}$$

where $p_k(+1|x) = 1/2 + 1/2\psi(x)$ is the probability that $x \in R_T(\Omega)$ based on the classifier, and $p_k(-1|x) = 1/2 - 1/2\psi(x)$ is the probability that $x \notin R_T(\Omega)$ based on classifier. The entropy would be large,if the sample x has uncertainty whether samples is in set or not, $\psi(x) \approx 0.5$

The AL algorithm terminates once N_s samples have been selected. The selection of samples is performed in a greedy manner, meaning that the algorithm selects the sample in each iteration that provides the most discerning information and minimizes overlap with the current labeled set. The classifiers at each iteration can be interpreted as the belief, based on the labeled set, that a state belongs to the reachable set.

The advantage of using AL methods over traditional supervised learning is the iterative improvement of learning performance by utilizing prior learners. Sampling is a cheap computational procedure compared to querying the oracle, and well-distributed samples ensure that informative samples exist near the boundary of the reachable set being estimated.

The mutual information between the labeled and unlabeled samples can be estimated by computing the relative distance between the distributions of the samples conditional upon a classifier . This mutual information is quantified by the mutual information function $J_D(S)$. Here is the definition:

$$J_D(S) = \frac{1}{|L_k|} \sum_{i=1}^{|L_k|} max D_{ij}^k$$

This mutual information can be estimated by computing the relative distance between the distributions of the ith and jth samples conditional upon the classifier ψ_k.

In the paper it employs the Kullback-Liebler divergence metric to calculate mutual information score, however in the implementation we have used the different method to get the information score with the help of `sklearn.metric` library which has `mutual_information_score` function, which does the same task.

The B most useful samples in S are obtained with solving $argmax$ of the sum of $J_E(S)$ and $J_D(S)$.

$$L'k = argmax J(S_0)$$
where,
$$J(S_0 = J_E(S_0) + J_D(S_0)$$

The informativeness-redundancy trade-off problem is solved using a cardinality constrained submodular maximization. The goal is to select the most informative samples from the set S by taking into consideration both the entropy of the samples and their mutual information with the prior labeled samples in L_k. The entropy component, J_E, promotes the selection of samples that provide new information, while the mutual information component, J_D, discourages the selection of samples that are redundant with respect to the prior labeled samples. Empirically, the addition of J_D does not always have a significant impact on the quality of learning, and can be considered an optional component of the algorithm.

The optimization of the given problem can be achieved efficiently through a greedy algorithm that iteratively chooses the sample x from S that has the greatest impact on J until a total of B samples have been selected.

Here is what pseudocode of greedy algorithm looks like:

Algorithm 1 Greedy

Require: Set functions J_E, J_D
Require: Unlabaled set, S
Require: Batch size, B
Ensure: Actively learned samples, $L'_{B,G}$
  $L'_{k,G} \leftarrow \emptyset$
  for $k = 1 : B$ **do**
    $x^* \leftarrow argmax_{x \in S \setminus L'_{k,G}} \Delta(x|L'_{k,G}; J)$
    $L'_{k,G} \leftarrow L'_{k,G} \cup \{x^*\}$
  end for
  return $L'_{B,G}$

The time complexity of the given algorithm has $\mathbf{O}(|S|B^2)$

In python this code would look like this:

```
def greedy_active_learning(labeled_samples, unlabelled_samples,batch_size):
    # Print the initial labeled samples
    print("initial_labeled_samples:",labeled_samples)
    # Initialize an empty list to store J values
    J_values = []

    for k in range(0,batch_size):
        for i in range(0,len(unlabelled_samples)):
            # Get the sample from the unlabelled samples
            sample = unlabelled_samples[i]
            # Append the J value for this sample to the J_values list
            J_values.append(calculate_JE(sample, labeled_samples)+calculate_JD(sample,labeled_samples))
            if not J_values:
                break

        # Find the argmax of J as given on paper
        idx = J_values.index(max(J_values))
        print("index_of_the_most_informative_sample_on_unlaballed_set:",idx)
        # Get the new sample from the unlabelled samples
        new_sample = unlabelled_samples[idx]
        # Append the new sample to the labeled samples
        labeled_samples = np.append(labeled_samples, [new_sample], axis=0)
        # Remove the labeled sample from the unlabelled samples
        unlabelled_samples = np.delete(unlabelled_samples, idx, axis=0)
        # Clear the J_values list for the next iteration
        J_values.clear()

    return labeled_samples
```

To ensure the unlabeled samples are well dispersed through out X, the size of samples should be large. As, first algorithm exhibits time complexity of $\mathbf{O}(|S|B^2)$ increasing cardinality of S and B could increase execution time massively. To tackle this problem, random subsampling and distributed computations can be exploited. We present two variants: (i) a stochastic variant of Algorithm 1 to facilitate active learning on large sets of unlabeled samples with time complexity linear in $|S|$ and B; and, (ii) for cases when $Nc \geq 2$ clusters are available for distributed implementation, we provide a distributed algorithm..

3.2 Stochastic Greedy Active Learning

3.2.1 Definition of Stochastic Greedy Algorithm

Stochastic greedy algorithm is a heuristic optimization method that combines the greedy approach of making the best decision at each step with the random sampling of choices.[6] The algorithm starts by generating a random subset of the

available choices, then evaluates these options to determine the best decision based on the objective function. The best decision is then selected and added to the solution, and the process repeats until a stopping criteria is met.

This method is used in various optimization problems, including combinatorial optimization and machine learning, as it provides a fast and efficient solution while also avoiding being trapped in local optima. The random sampling component of the algorithm helps to ensure that it is not susceptible to becoming stuck in poor solutions, as it increases the chance of exploring different options and discovering better solutions.

3.2.2 Algorithm Principle

This algorithm instead of iterating through all samples in S and computing the marginal gain, it selects a random subset from this unlabeled samples containing q elements of S. The another reason that this method is effective is that, for a given $\epsilon > 0$, and choosing q large enough, we can prove that the selected subset will contain elements that is gathered in the previous greedy algorithm, with probability of $1 - \epsilon$.

Here is the pseudocode of Algorithm:

Algorithm 2 Stochastic Greedy

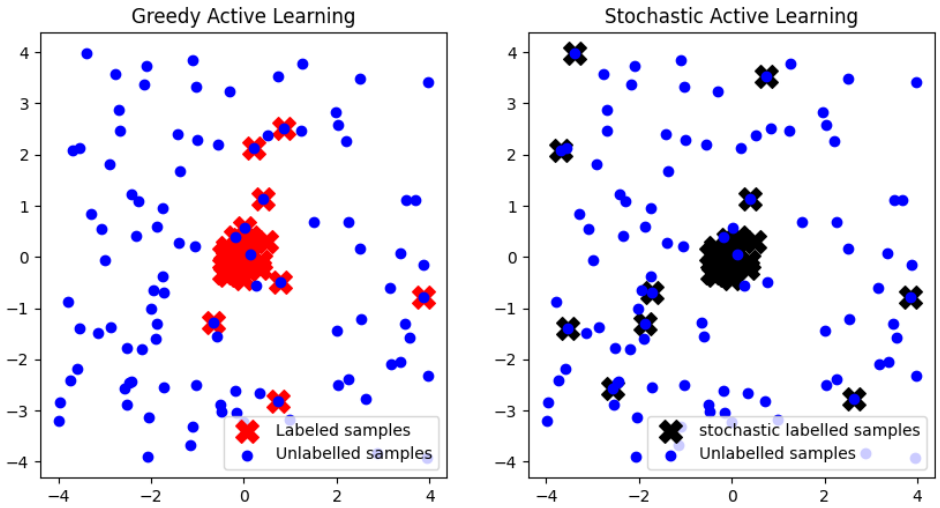
Require: Set functions J_E, J_D
Require: Unlabeled set, S
Require: Batch size, B
Require: Approximation tolerance, $epsilon$
Ensure: Actively learned samples, $L'_{B,SG}$
  $q \leftarrow (|S|/B)ln(1/\epsilon)$
  $L'_{k,G} \leftarrow \emptyset$
  for $k = 1 : B$ **do**
    $S_q \leftarrow$ random subset of $|S|/L'_{k,SG}$ with q elements
    $x^* \leftarrow argmax_{x \in S \setminus L'_{k,G}} \Delta(x|L'_{k,SG}; J)$
    $L'_{k,SG} \leftarrow L'_{k,SG} \cup \{x^*\}$
  end for
  return $L'_{B,G}$

and the python implementation of this would look like something like this:

```
def stochastic_greedy_active_learning(labeled_samples, unlabelled_samples,batch_size,epsilon):
# Print the initial labeled samples
print("initial labeled samples:",labeled_samples)
# Initialize an empty list to store J values
J_values = []
q = int((len(unlabelled_samples) / batch_size) * math.log(1 / epsilon))

for k in range(0,batch_size):
    unlabelled_samples_Q = []
    unlabelled_samples_Q = random.sample(list(unlabelled_samples), q)
    for i in range(0,len(unlabelled_samples_Q)):
        # Get the sample from the unlabelled samples
        sample = unlabelled_samples_Q[i]
        # Append the J value for this sample to the J_values list
        J_values.append(calculate_JE(sample, labeled_samples)+calculate_JD(sample,labeled_samples))
        if not J_values:
            break

    # Find the argmax of J as given on paper
    idx = J_values.index(max(J_values))
    print("index of the most informative sample on unlaballed set:",idx)
    # Get the new sample from the unlabelled samples
    new_sample = unlabelled_samples[idx]
    # Append the new sample to the labeled samples
    labeled_samples = np.append(labeled_samples, [new_sample], axis=0)
    # Remove the labeled sample from the unlabelled samples
    unlabelled_samples = np.delete(unlabelled_samples, idx, axis=0)
    # Clear the J_values list for the next iteration
    J_values.clear()

return labeled_samples
```

4 Results

The dataset we are using is for exemplary purposes and does not represent the actual reachable set of some dynamic system. Dataset is created randomly within the given boundries and the initially labelled samples are also like that. The stability of results are not very adequate, as this code is not fully deployed. However, here are some results of given algorithms.



The blue samples that have ”X” under them are the samples that has been selected.

5 Conclusions

In conclusion, this paper has presented the implementation of the Active Learning method for estimating reachable sets of systems with unknown dynamics in Python. The implementation is based on the Stochastic Greedy Algorithm and the Greedy Algorithm, both of which are well-established techniques in the field of Active Learning.

Although, the implementation is not fully stable, the results on the original paper demonstrate that the Active Learning method is able to efficiently and effectively estimate the reachable sets of unknown dynamic systems. The implementation of the algorithms in Python has made it possible to get the entry level information of developing, and could be fully developed to apply to a wide range of systems, making it a valuable tool for system analysis and control.

References

[1] Zhang, X. Wu, and V. S. Shengs, “Active learning with imbalanced multiple noisy labeling,” IEEE Transactions on Cybernetics, vol. 45, no. 5, pp. 1095–1107, 2014.

[2] Cushman, R. H., & Langer, R. S. (1997). Dynamical Systems. Springer Science & Business Media.

[3] Settles, B. (2010). Active learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, 4(1), 1-114.

[4] K. J. Åström and B. Wittenmark, ”Computer-Controlled Systems: Theory and Design,” Prentice Hall, Englewood Cliffs, NJ, 1984.

[5] L. Zhang and T. A. Johansen, ”Reachability Analysis of Hybrid Systems: A Survey,” in Proceedings of the IEEE, vol. 104, no. 10, pp. 1765-1782, Oct. 2016.

[6] Chong, E. K. P., & Zak, S. H. (2007). An Introduction to Optimization. John Wiley & Sons.