An Active Learning Method for the Comparison of Agent-based Models

Swapna Thorve[1], Zhihao Hu[2], Kiran Lakkaraju[3], Joshua Letchford[3], Anil Vullikanti[1], Achla Marathe[1], Samarth Swarup[1]
[1]University of Virginia,
[2]Virginia Tech,
[3]Sandia National Lab,
{st6ua,vsakumar,achla,swarup}@virginia.edu,huzhihao@vt.edu,{klakkar,jletchf}@sandia.gov

ABSTRACT

We develop a methodology for comparing two or more agent-based models that are developed for the same domain, but may differ in the particular data sets (e.g., geographical regions) to which they are applied, and in the structure of the model. Our approach is to learn a response surface in the common parameter space of the models and compare the regions corresponding to qualitatively different behaviors in the models. As an example, we develop an active learning algorithm to learn phase transition boundaries in contagion processes in order to compare two agent-based models of rooftop solar panel adoption.

KEYWORDS

agent-based modeling; active learning; response surface methods; model comparison

ACM Reference Format:
Swapna Thorve[1], Zhihao Hu[2], Kiran Lakkaraju[3], Joshua Letchford[3], Anil Vullikanti[1], and Achla Marathe[1], Samarth Swarup[1]. 2020. An Active Learning Method for the Comparison of Agent-based Models. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), Auckland, New Zealand, May 2020,* IFAAMAS, 9 pages.

1 INTRODUCTION

Complex large-scale agent-based models (ABM) are becoming increasingly common, in a huge number of application areas, such as public health, infrastructure systems such as transportation and power, disaster evacuation, and technology adoption. ABMs are designed to answer specific questions within an application, and their design is data-driven. As a result, there can be multiple ABMs with a similar overall structure, but differences in the specific model components, their interactions and parameters. This raises the general question of how to compare such models [2, 6]. Axtell et al. [2] were the first to address this question, and developed the "docking" technique, which involves verifying whether or not the dynamical properties of one ABM can be regenerated by another. However, this is computationally intractable as ABMs become complex, and a very restrictive notion. Other notions of validation and equivalence of ABMs have also been explored [e.g., 4, 22]. However, these only allow restricted forms of comparison between ABMs, and do not provide efficient computational tools for comparison based on specific characteristics in the phase space.

At an abstract level, these questions are analogous to the many notions of phase space equivalence of dynamical systems [1, 18]. The approach of [2] attempts to compare precise structural properties in the phase space, which is NP-hard, in general [1]. In this paper, we present a general and more scalable framework to make these types of comparisons between ABMs, based on comparing approximate representations of the phase spaces of the ABMs; specifically, we consider the structure of the region which correspond to "phase transitions", i.e., where the system has very different behavior by small change in parameters. While this does not correspond to exact phase space equivalence, this notion can give useful insights in many applications where the ABMs work on different domains.

As a specific example, we consider ABMs for adoption of rooftop solar panels by households in three different regions of the United States. A question of interest for local governments and utilities is to understand who will adopt solar, and how to increase adoption. We compare two different ABMs, one developed for California [24], and the other for Virginia that we present here, based on a model presented earlier by Hu et al. [11]. The probability of adoption by a household depends on a number of factors, including demographics and characteristics of the house, as well as *peer effects*, captured by the number of households who have adopted within a 1-4 mile range. The two models have a large fraction of common factors, but some which are distinct, e.g., pool ownership, which is a factor in the ABM of [24], but not in that of [11]. The datasets used in the calibration of these two ABMs have different characteristics, with a much larger adoption rate in California. In order to compare these two ABMs, we study whether it is somehow easier to have a large number of adoptions in one region than another, or whether it is just a chance difference. Our contributions are summarized below.

- We design a general methodological framework based on the response surface method. We introduce a notion of *characteristic distribution* of an ABM in terms of phase space properties as the probability distribution of the ABM outcomes within some range. A specific example we consider is a range within which the dynamics exhibits a phase transition. We quantify the *disagreement* between two ABMs through the differences in these probability distributions.
- We design a machine learning approach based on active learning for estimating the characteristic distribution. Active learning helps us reduce the number of times the simulation

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 2020, Auckland, New Zealand.

has to be run. Thus, this is a much more scalable approach for complex ABMs.
- We illustrate our approach on the two ABMs for comparing solar adoption. Our results show that it is indeed easier to see large numbers of adopters in the San Diego model, but that there are significant differences between the two regions of Virginia that we study also.

Organization. The rest of this paper is organized as follows. First we present the general methodological framework (Section 2). Then we describe the two models we are comparing (Section 3). After describing this method in detail (Section 4), we present results from computational experiments with the two models (Section 5). We end with a discussion of related work and future directions.

2 FRAMEWORK

Response surface methodology [5, 7], also known as metamodel-based methods [3] for stochastic simulations is generally used for optimization [19] and calibration [9, 13]. The general idea is to approximate the stochastic objective function (the "response") by a low order polynomial function of the independent variables over a part of the domain. RSM typically runs in phases. The process starts with a screening experiment, which identifies a subset of candidate variables, which are most significant in the region of interest. Next, a first-order model is used to approximate the response, and a line search is used to find an improving direction for the objective. In the second phase, a second-order model is used to approximate the objective, since usually a response surface has curvature near the optimum.

We denote the agent-based simulation model as a stochastic function, $F(\xi_1, \xi_2, ..., \xi_k)$, of its parameters, assuming a fixed initial condition. In response surface methodology, we generally fit the expected value of this function,

$$f(\xi_1, \xi_2, ..., \xi_k) = \mathbf{E}(F(\xi_1, \xi_2, ..., \xi_k)), \tag{1}$$

where F is the stochastic output, given parameters $\xi_1, \xi_2, ..., \xi_k$, and $\mathbf{E}$ denotes expectation. This is appropriate when the goal is optimization or calibration.

However, when we are using the ABM to model a specific observed phenomenon, as is the case in the scenarios described in Section 3, the real-world data represent only one stochastic realization of the model. Therefore, instead of taking the expectation, we characterize the behavior of the ABM in terms of the probability of seeing a particular output given a particular parameter setting. For ease of exposition, we assume that the simulation outputs one continuous variable, y, though the formalism generalizes straightforwardly to multiple outputs. We relate y to the parameters as follows.

$$P(y_{low} < y < y_{high}) = \int P(y_{low} < F < y_{high}|\Xi)P(\Xi), \tag{2}$$

where $\Xi = [\xi_1, \xi_2, ..., \xi_k]$, and $P(\Xi)$ is a prior probability over the parameter space. Thus, the ABM can be characterized as a discretized probability distribution, using a set of bins denoted by their bin boundaries, $\{[y_0, y_1], [y_1, y_2], ..., [y_{n-1}, y_n]\}$. The choice of bins depends on the domain of the model. For example, models of contagion processes exhibit sharp transitions, which are a natural choice for bin boundaries in that case, as we will see in the experiments section. We refer to this distribution as the *characteristic distribution* for the ABM.

We define the *characteristic distance* between two ABMs as the distance between their characteristic distributions.

$$d(F_1, F_2) := D(P_1(y), P_2(y)). \tag{3}$$

There are multiple valid choices for D, such as the (symmetric) KL-divergence, mean-squared distance, total variation distance, earth-mover's distance, etc. Note that this distance is well-defined even if the two ABMs have entirely different parameter spaces, since it is defined only over the output space. Thus it is a fairly general method of comparison. For a given observed value, y_{obs}, we can also directly compare the probabilities assigned by the two models to the corresponding bin.

$$d_{obs}(F_1, F_2) := P_1(B_{obs}) - P_2(B_{obs}), \tag{4}$$

where B_{obs} is the bin within which y_{obs} lies. This directly tells us how much more likely it is to see y_{obs} in one model versus the other.

For a general comparison of the two ABMs in the case where the two ABMs have an overlap in their parameter space (i.e., they have some parameters in common), we can have a more detailed comparison. Let Ξ_c be the parameters that the two ABMs have in common. We can partition this subspace of the parameter space into regions based on the most likely bin for y for each parameter setting.

$$B(\Xi_c) = \arg\max_B \int_{\Xi \setminus \Xi_c} P(B|\Xi)P(\Xi \setminus \Xi_c), \tag{5}$$

where B denotes a bin, corresponding to the bin boundaries defined earlier, and $\Xi \setminus \Xi_c$ denotes the parameters other than the common parameters. The equation above assigns to each point in the common parameter subspace, a bin corresponding to the most likely output at that point.

Now we define the *disagreement*, Δ, between the two ABMs as the probability of choosing a parameter setting, according to the prior distribution, that results in a difference in the outputs of the two models.

$$\Delta(F_1, F_2) = \int_{\Xi_c} (1 - \mathbf{1}(B_1(\Xi_c), B_2(\Xi_c)))P_1(\Xi_c), \tag{6}$$

where $\mathbf{1}(B_1(\Xi_c), B_2(\Xi_c))$ is an indicator function that is 1 if $B_1(\Xi_c) = B_2(\Xi_c)$ and 0 otherwise. Δ gives the total probability, over the subspace Ξ_c, that the outputs of the two ABMs will fall into different bins. Note that Δ is a directed measure, since $\Delta(F_1, F_2) \neq \Delta(F_2, F_1)$.

There are various ways to compute the integral in equation 5. The typical approach in simulation science is to use adaptive experiment designs [21]. Here we propose a machine learning approach based on active learning. The general idea is to train a multi-class classifier, where a class corresponds to a bin, for each model. Since the simulations can often be expensive to run, an active learning approach can help minimize the number of times the simulation has to be run. The classifiers are used to estimate $B(\Xi_c)$ for each ABM. Once the classifiers have been trained, we can use them to estimate Δ.

The active learning approach to training the classifier involves running the simulation in a loop with the classifier. In each round, the simulation is run to generate additional labeled points for the

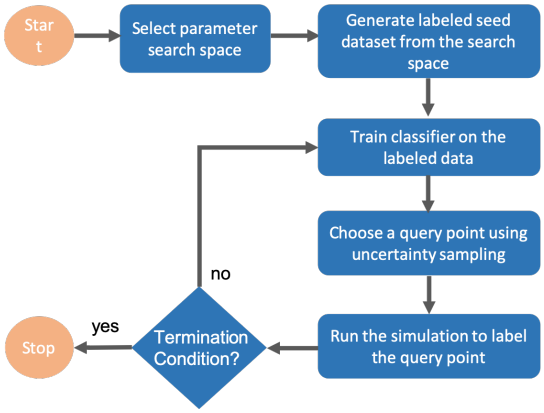


Figure 1: Overview of the presented methodology - A common set of parameters is chosen from both ABMs and an active learning framework is implemented to learn the decision boundary that separates the bins.

classifier. Then the classifier is trained on the updated data set and we do uncertainty sampling [14] in the parameter space to generate new parameter settings where the simulation is to be evaluated in the next round. The process stops when the labels generated by the simulation agree with the labels generated by the classifier, as illustrated in Figure 1. In the particular scenario we study below, we will see that we can exploit domain semantics to generate multiple useful labeled points from one query. We continue below by first describing the models that we built for studying rooftop solar panel adoption in rural Virginia, USA, and the model we compare with, which was built to study solar panel adoption in San Diego, California, USA. Then we will describe how this framework is instantiated for these models.

3 ROOFTOP SOLAR PANEL ADOPTION

We compare two ABMs for rooftop solar adoption, one built by Zhang et al. for San Diego, California [24] and the other for Shenandoah Valley region in Virginia, as described below in Section 3.1. They both use a set of demographic, social, economic and geographical variables to assess the probability of adoption for each household in respective study areas. A logistic regression model is built in each case to identify important factors that influence solar adoption. These factors are then used by their respective ABMs to study the diffusion of adoption. Since the solar penetration rate in Virginia is much smaller compared to California, a decision-adjusted logistic regression model was used to handle the issue of class imbalance in Virginia.

3.1 The Virginia Model

We build a statistical model to identify features that contribute to a household's decision to adopt rooftop solar panels. However, due to low penetration of solar adopters in rural regions, the data on solar adopters is sparse, which makes it difficult to build a good prediction model. Given highly imbalanced training data, traditional statistical methods tend to predict most households to be non-adopters in order to minimize the mis-classification error and provide high overall prediction accuracy.

Table 1: Virginia model coefficients.

Feature	Description	Coefficient
acreage	Acreage of the house.	-0.123
area_type	Rural (0) or urban (1).	-1.79
asrYear	Year house was built.	0.00184
bedrooms	Num of bedrooms.	0.0881
PubCold/Very Cold	Type of climate.	-1.09
Hot-Humid	Type of climate.	0.00883
daily Consumption	Avg. energy used in Wh	5.65e-07
edu2	High school diploma	-0.0117
edu3	Some college or Associate's degree	0.332
Propane	Type of fuel used for heat.	-0.528
Fuel oil or kerosene	Type of fuel used for heat.	0.269
Wood	Type of fuel used for heat.	0.472
numCarStorage	Num. of car storage in the house.	0.331
swimpool	Pool present or not.	0.169
totalVal	Estimated house value.	-3.66e-07
totalValI	Indicator value for totalVal	-1.12
Single-family detached house	House type	0.00162
Single-family attached house	House type	-0.479
householdSize	Family size.	0.123
Mile1	Adopters within 1 mile.	0.399
Mile2	Adopters within 2 mile.	0
Mile3	Adopters within 3 mile.	0.0299
Mile4	Adopters within 4 mile.	0.0376

In our study, we are more interested in identifying potential adopters so we apply a decision-adjusted modeling approach from Mao et al. [16], and Hu et al. [11]. The decision-adjusted approach optimizes the prediction model with respect to a user-stated evaluation criterion. We set this criteria to maximize the sum of True Positive Rate and True Negative Rate. The decision-adjusted approach can be applied to different statistical models, here we choose logistic regression model as our baseline model.

The decision-adjusted solar adopter prediction model is as follows. Suppose a prediction model is obtained from the data set $\{X_i, I_i(\delta), Y_i\}_{i=1}^n$, where X_i are the demographic features as shown in Table 1. Y_i is the response, $Y_i = 1$ means household is an adopter and $Y_i = 0$ means non-adopter, $I_i(\delta)$ are the indicator features for corresponding demographic features, which is defined as,

$$I_i(\delta) = \begin{cases} 1, & \text{if } X_i > \delta; \\ 0, & \text{if } X_i \leq \delta, \end{cases}$$

The indicator features are introduced when the coefficients of linear combination are not able to capture all information in the model. For example, if the coefficient of a feature is positive, then a

larger value of the feature will increase the likelihood of adoption. However, this positive relationship may not be constant; it may be strong when the value of the feature is small, and weak when the value of the feature is large. The indicator features address this issue.

The logistic regression model has the link function below,

$$\Pr(Y = 1|X, I(\delta)) = \frac{exp\Big(\big(X, I(\delta)\big)^T \beta\Big)}{1 + exp\Big(\big(X, I(\delta)\big)^T \beta\Big)}$$

Where β is the regression coefficient. For the logistic regression with elastic net, β is estimated through maximizing a penalized log-likelihood $l(\beta) - P(\beta, \alpha, \lambda)$, where

$$P(\beta, \alpha, \lambda) = \lambda\big((1-\alpha)\frac{1}{2}||\beta||_2^2 + \alpha||\beta||_1\big)$$

The predicted label for this household is determined by the following, where τ is the threshold probability for determining whether a household is adopter or not.

$$\hat{Y}_i = \hat{Y}_i(X_i, I_i(\delta), \alpha, \lambda, \tau) = \begin{cases} 1, & \text{if } \Pr(Y_i = 1) \geq \tau \\ 0, & \text{if } \Pr(Y_i = 1) < \tau \end{cases}$$

Then the optimization problem for our decision-adjusted model is

$$\max_{\delta, \alpha, \lambda, \tau} \Omega\Big(\hat{Y}_i(X_i, I_i(\delta), \alpha, \lambda, \tau), Y\Big)$$

Where $\Omega(\cdot)$ is a function of predicted label $\hat{Y}_i(X_i, I_i(\delta), \alpha, \lambda, \tau)$, and true label Y. It is determined by the specific goals of the study, as defined by the user.

In the decision adjusted model, we want to maximize (Ω = TPR+TNR) given τ and δ. Here 'TPR' refers to true positive rate, 'TNR' refers to true negative rate, 'FP' refers to false positives, and 'FN' refers to false negatives. The TPR+TNR is defined as:

$$\Omega = \frac{TP}{TP + FN} + \frac{TN}{TN + FP} = \frac{TP}{Condition\ positives} + \frac{TN}{Condition\ negatives},$$

This is an improvement over accuracy as the performance metric, which is given by:

$$\text{ACCURACY} = \frac{TP + TN}{TP + FN + TN + FP} = \frac{TP + TN}{n}.$$

The tuning paraments selected from the decision-adjusted approach are:

- $\tau = 0.0022$, a household with predicted probability greater than 0.0022 is classified as an adopter.
- $\lambda = 0.000303$, $\alpha = 0.4$, which are values of tuning parameters in enet penalty.
- δ(totalVal) = 75000, if the value of feature totalVal is greater than 75000, then the value of the corresponding indicator feature is 1.

3.2 The San Diego Model

The San Diego model was developed by Zhang et al. [24]. The model was trained on extensive data collected to the California Solar Initiative https://www.gosolarcalifornia.ca.gov/about/csi.php. In addition, property assessment data for San Diegor county and electricity utilization data for participants in the rebate program was collect (energy utilization data as limited to the 12 months before adoption). The data set spanned 6 years and 8500 adopters and included detailed information about the solar panel purchasing decision, including the system size, reported cost, incentive, whether the system was purchased or leased and date of adoption.

Table 2: List of features in the San Diego model

Feature	Description
ownerocc	Owner occupied (binary)
Ls	Lease option available (binary)
wt	Winter (binary)
st	Spring (binary)
sm	Summer (binary)
fracInstall	Installation density in zipcode
NPV	NPV (Purchase)
Mile1	Installations within 1 mile radius.
Mile1/4	Installations within one fourth mile radius.

The ABM developed from this data used machine learning techniques to train an individual model of solar panel adoption behavior. Individual agents changed their behaviors based on household characteristics, seasons and peer-effects. [24] provides details on the specific variables used and the results. Table 2 summarizes the variables of the model.

For this work we used the model provided here: https://github.com/haifeng-zhang/ddabm-solar which focused on modeling a single zip code within the San Diego region.

4 MODEL COMPARISON METHOD

In this section, we instantiate the framework described in Section 2 to compare the models described in Section 3. Both the Virginia model and the San Diego model are network contagion models, where a contagion (in this case a technology: rooftop solar panels) spreads through a network. Both models belong to the general class of *SI* contagion models, drawn from mathematical epidemiology, where *S* stands for *Susceptible* and *I* stands for $Infectious$. In our case, $Infectious$ corresponds to adopter. Once a household has adopted solar panels, their peer influence on their neighbors is assumed to persist indefinitely. This means that, in the limit, all households in both models will be adopters, as long as the parameters are set in such a way that the probability of adoption if at least one neighbor has adopted is non-zero. However, depending on the parameter settings, it can be the case that the probabilities of adoption are so low that we see very few adoptions in the duration for which the simulations are run. Network structure can also play a role in speeding up or slowing down the spread of the contagion through the network.

As the probability of adoption increases, the SI model undergoes a phase transition, where the simulation shows a sharp qualitative change in its behavior. As the probability of adoption crosses a threshold, the simulation quickly changes from only a few nodes being adopters to a lot (or most) of the nodes becoming adopters in a short amount of time. Due to this qualitative behavior, we choose just two bins to describe each simulation in Section 3, which we refer to as B_0 and B_1, corresponding to small and large numbers of adopters, respectively. The actual values chosen are listed in Section 5.

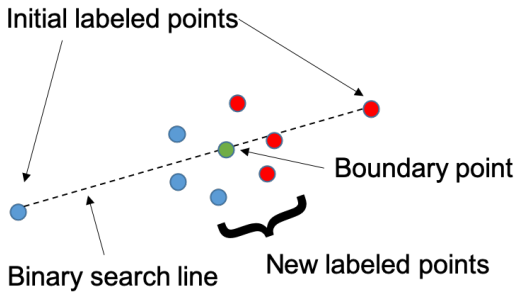


Figure 2: A schematic illustration of the binary search process. Blue points are in B_0, red are in B_1, and the green point is a boundary point.

Another nice aspect of this scenario is that the variance (or standard deviation) in the output of the simulation shows a sharp peak at the phase transition boundary, and tends to be low away from the boundary. This is illustrated in Figure 3. Thus, by running multiple runs of the simulation for any chosen point in parameter space, we get a clear signal if the point is close to the phase transition boundary. This scenario is unique in the sense that we can actually know where the decision boundary is for the classification algorithm, which is not the case in typical machine learning scenarios. We make use of this fact in our active learning algorithm below.

With a slight abuse of notation, we write $\Xi \in B_0$ if, for the point Ξ in the parameter space, $B(\Xi) = B_0$. The basic idea of the algorithm is that if we have two points, $\Xi_0 \in B_0, \Xi_1 \in B_1$, we can do a binary search in the parameter space along the line between Ξ_0 and Ξ_1 to find a point on the phase transition boundary (a "boundary point") by observing where the standard deviation peaks. This requires doing multiple runs of the simulation for each evaluated point in the parameter space, but these runs can be done in parallel to save time. This binary search process is given in Algorithm 1.

Since the phase transition boundary is quite sharp, once we find a boundary point, we can generate k other nearby points randomly around it at a small distance, ϵ, and label them using the simulation. Some of these will fall in B_0 and some in B_1. Since they are close to each other, they will strongly constrain the decision boundary for the classifier. Thus, given two initially labeled points Ξ_0 and Ξ_1, we can generate k useful labeled points for the classifier. Figure 2 schematically illustrates this process.

We follow this by training the classifier on the labeled points generated so far. At the end of r rounds, we have rk labeled points in the training set. In order to start round $r+1$, we do uncertainty sampling with the trained classifier at the end of round r, by generating a point on the classifier decision boundary that is far from the training set. This represents a point in the parameter space about which we have a high degree of uncertainty. We run the simulation to label this point. If it falls within B_0, we choose an already labeled point in B_1 (typically the farthest one), or vice versa, to initialize the binary search in round $r+1$.

We can define two different stopping criteria. First, if we have a budget on the total number of runs of the simulation, K, then we stop after round r if going to round $r+1$ would cause the number of simulation runs to exceed K, i.e., if $(r+1)k > K > rk$. Alternatively, as the active learning proceeds, we should find that when we choose points on the decision boundary of the classifier using uncertainty sampling, they actually turn out to fall on the phase transition boundary according to the simulation. This means that the classifier is approximating the phase transition boundary well, and we can stop training. The overall algorithm for the active learning procedure is given in Algorithm 2.

Once training is complete, we compute the characteristic distance and the disagreement between the models by sampling. We generate a large number of points according to the prior distribution, $P(\Xi)$, in the parameter space and label them using the trained classifier. If N is the total number of points generated, and N_0 is the number that are labeled as being in B_0, the characteristic distribution is $(N_0/N, (N-N_0)/N)$. To calculate the disagreement, we have to count the number of points, N' that are labeled differently by two models. Then the disagreement is, $\Delta = N'/N$.

Algorithm 1 Binary search to find a boundary point

```
1:  procedure BinarySearch(pt1, pt2)
2:      [pt1Mean, pt1Stdev] = RunDiffusionModel(pt1)
3:      if pt1Stdev ≥ θsd then
4:          Add pt1 to boundaryPoints
5:          return [pt1, pt1Mean, pt1Stdev]
6:      else
7:          Add [pt1, pt1Mean, pt1Label] to evaluatedPoints
8:      end if
9:      [pt2Mean, pt2Stdev] = RunDiffusionModel(pt2)
10:     if pt2Stdev >= θsd then
11:         Add pt2 to boundaryPoints
12:         return [pt2, pt2Mean, pt2Stdev]
13:     else
14:         Add [pt2, pt2Mean, pt2Label] to evaluatedPoints
15:     end if
16:     m = (pt1 + pt2)/2.0
17:     while m ∉ boundaryPoints and pt1Label ≠ pt2Label do
18:         [mMean, mStdev] = RunDiffusionModel(m)
19:         if mStdev >= θsd then
20:             Add m to boundaryPoints
21:             return [m, mMean, mStdev]
22:         else
23:             Add [m, mMean, mLabel] to evaluatedPoints
24:         end if
25:         Assign m to pt1 or pt2, s.t. pt1 and pt2 have different
    labels
26:         m = (pt1 + pt2)/2.0
27:     end while
28: end procedure
```

5 EXPERIMENTS

Experiments are done with two agent based models in three regions with different population sizes. Numbers of agents in each of the regions are as follows: Rappahannock county has 2495 agents, San Diego has 12925 agents, Shenandoah Valley Region (SVR) has 138043 agents.

Algorithm 2 Active learning for predicting decision boundary

Input : $DiffusionModel$, 2D search space for parameters $p1, p2$
Output : $evaluatedPoints$, $boundaryPoints$
procedure LearnDecisionBoundary
  $start$ = $[p1Min, p2Min]$
  end = $[p1Max, p2Max]$
  $[bPt, bMean, bStdev]$ = BinarySearch$(start, end)$
  EvaluateNearbyPoints(bPt) Add these points to $evaluatedPoints$
  $noRounds$ = 1
  while $noRounds$ <= 5 **do**
    nPt = GetNextPointViaActiveLearning
    $[nPt, nMean, nStdev]$ =RunDiffusionModel(nPt)
    if $nStdev >= \theta_{sd}$ **then**
      Add nPt to $boundaryPoints$
    else
      Add $[nPt, nMean, nLabel]$ to $evaluatedPoints$
    end if
    $oppPt$ = Another point with label opposite to $nLabel$ and farthest from nPt
    $[bPt, bMean, bStdev]$ = BinarySearch$(nPt, oppPt)$
    EvaluateNearbyPoints(bPt) Add these points to $evaluatedPoints$
    $noRounds++$
  end while
  return $evaluatedPoints$, $boundaryPoints$
end procedure

Both the agent-based diffusion models simulate the number of households adopting rooftop solar. The model presented in Section 3.1 is used for Rappahannock county in Virginia and Shenandoah Valley Region in Virginia. The model presented in Section 3.2 is used for San Diego. In order to facilitate comparison between the 2 ABMs, we choose to explore the common parameters in both the models: Mile1 and NPV (see Tables 1 and 2). All experiments are conducted in 2D space using Mile1 and NPV. This makes it easy to visualize the results, as we show in Section 6. Thus, our prior distribution, $P(\Xi_c)$, is chosen to be a uniform distribution over a rectangular region of the common parameter space defined by the chosen ranges for Mile1 and NPV. In order to simplify calculations, we also assume fixed values for the other parameters in each model, as given by the regression coefficients in Section 3, instead of integrating over them, as defined in the framework in Section 2.

We define a point in the parameter space to be a boundary point if the standard deviation of the number of adopters generated by the simulation for that parameter point is higher than a threshold, θ_{sd}. If the point is not a boundary point, then it is labeled as being in B_1 if the mean number of adopters is greater than a threshold, θ_m. Otherwise it is assumed to be in B_0. In order to choose the thresholds, θ_{sd} and θ_m, we do a preliminary set of runs along the main diagonal of the chosen rectangular region in parameter space. Figure 3 shows the output of the Virginia model on Rappahannock data in terms of mean and standard deviation. Based on this, we set θ_{sd} to 100 and θ_m is set to 1000 for Rappahannock. Similar experiments are performed for SVR and San Diego regions to set

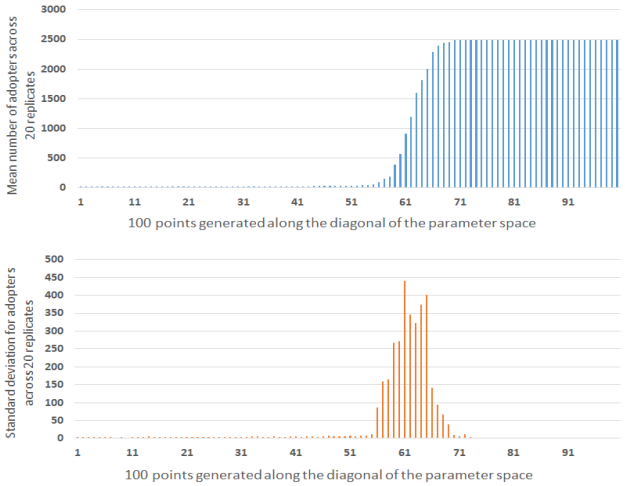


Figure 3: Mean and standard deviation of the number of adopters generated by the Virginia model for Rappahannock county along the diagonal of the chosen parameter space.

the thresholds. In all the experiment settings and results shown, the ABM results are averaged over 20 replicates to calculate mean and standard deviation. Table 3 shows the chosen thresholds for the three regions.

Table 3: Thresholds for evaluating unlabeled instances.

Regions	mean-threshold, θ_m	std-threshold, θ_{sd}
Rappahannock	1000	100
SVR	12000	3500
San Diego	120	12

Once the thresholds are set, we conduct three sets of experiments using the active learning method described above:

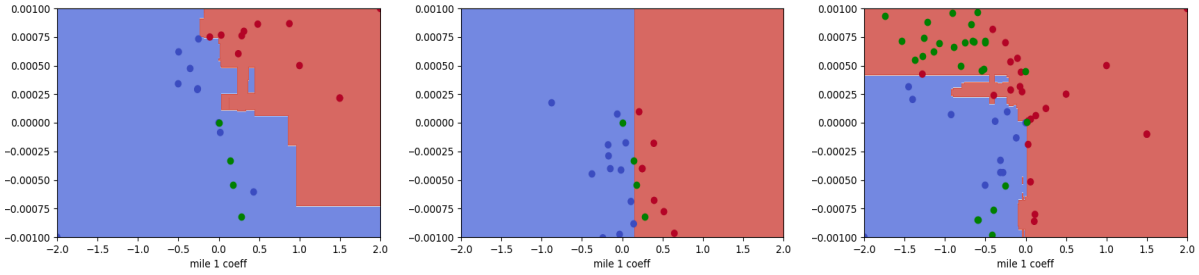
(1) Model 3.1 for Rappahannock county
(2) Model 3.1 for SVR region
(3) Model 3.2 for a zipcode in San Diego

We use random forest classifiers to learn the phase transition boundaries. Results are presented next.

6 RESULTS

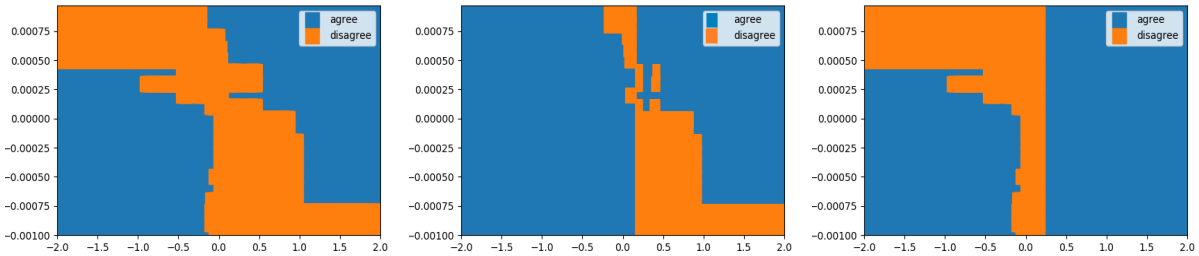
The learned decision boundaries are shown in Figure 4 for each of the three regions. All the evaluated points are plotted as well. The blue points show small numbers of adopters, the red ones show large numbers of adopters, and the green points are boundary points. We see that there are significant differences between the three regions. The blue area is largest in Rappahannock and smallest in San Diego.

This is made precise in Figure 6, which shows the characteristic distributions of the models for the different regions. Since we chose a uniform prior distribution, the heights of the bars correspond to the blue and red areas in Figure 4. B_0 (blue) represents small number of adopters and B_1 (red) represents large number of adopters.



(a) Rappahannock (b) SVR (c) San Diego

Figure 4: Decision boundary discovered by the active learning algorithm in the 2D parameter search space for Rappahannock, SVR and San Diego regions. The blue region (labeled as 0) represents a small number of adoptions and the red region (labeled as 1) represents a large adoptions. The x-axis is the range of mile1 feature coefficient and y-axis represents range of the NPV feature coefficient.



(a) Rappahannock and San Diego (b) Rappahannock and SVR (c) SVR and San Diego

Figure 5: Disagreement between ABMs

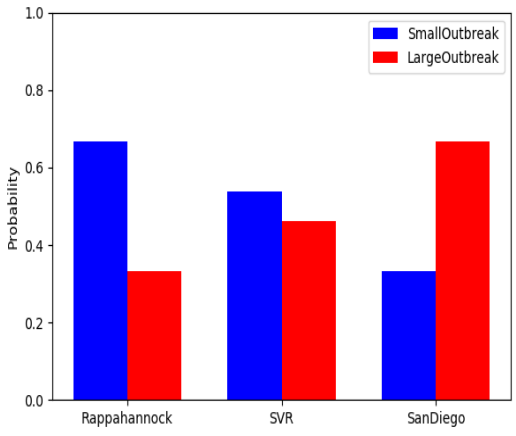


Figure 6: Characteristic Distributions of ABMs for Rappahannock, SVR and San Diego regions.

We see that small numbers of adopters are much more likely in Rappahannock than in San Diego, while SVR lies inbetween. In order to calculate the distance between characteristic distributions of the models, we will use Equation 3, where D is the total variation distance. Table 4 shows the pairwise distances between the models.

Next we compute the pairwise disagreement values for the models. As described earlier, this is done by generating a large grid of points (since we chose a uniform prior) and counting the number of points for which the two models disagree on the label. The results are shown in Table 5. We see that Rappahannock and San Diego have the largest disagreement, and SVR has a smaller disagreement with each of them. This matches what we find for characteristic distance, though it is possible for the two measures not to agree. The disagreement plots are shown in Figure 5. They show the region in the parameter space where the models produce different results, which gives a much more precise picture of the differences between the models. Note that, although we refer to the disgreement between the models, these differences are due to the data for Rappahannock and SVR, since the model is the same for those two regions. Whereas when we compare either of those with the San Diego model, the differences are due to a combination of data and model.

If two models have exactly the same parameters and model structure, then this method can be used to isolate the differences in outputs due to the differences in the data, which may be due to differences in the distributions of various features or due to

Table 4: Characteristic distance: Pairwise distances between the characteristic distributions, using total variation distance.

u	v	**TVD**
Rappahannock	San Diego	0.3338
Rappahannock	SVR	0.1269
San Diego	SVR	0.2069

differences in network structure. However, even if two models don't have exactly the same parameters and structure, we can still do a meaningful comparison, as we do for San Diego in comparison with either Rappahannock or SVR, though we cannot isolate the effects of the data alone.

Table 5: Disagreement: Rappahannock and SVR have the least disagreement whereas Rappahannock and San Diego have the largest disagreement.

Pair	**% of disagreement**
Rappahannock and San Diego	32.4%
Rappahannock and SVR	17%
San Diego and SVR	18.6%

7 RELATED WORK

Earlier researchers have considered comparison of models and simulations to increase confidence in results and in interpretation of the models, e.g. [2, 6, 23]. Work by [23] compares RePast [8] and Swarm [17] based simulations of four different social network models, i.e. random graphs, preferential attachment, and preference attachment with constant fitness, and with dynamic fitness. Structural properties of these networks were used to dock RePast and Swarm simulations. The results showed that docking could help validate a simulation and help migrate a simulation from one software package to another. Authors in [20] used Mathematica, Swarm and RePast to simulate a Beer distribution game. Louie et al. [15] show how model comparison and model alignment can help compare and contrast models, clarify assumptions and understand semantic differences in data usage.

8 CONCLUSION

We have presented a new methodology for comparing agent-based models. In fact, our approach applies to simulations in general, since we are not making explicit use of the fact that these are agent-based. We treat the simulation as a black box with a given parameter space, a given output, and a fixed input. We created a framework for comparison based on two new quantities we have defined: the characteristic distance and the disagreement.

We also presented a new agent-based model of rooftop solar panel adoption in rural Virginia, USA, and compared this model with an earlier model of rooftop solar panel adoption in San Diego, California, USA. We instantiated our framework to compare these models using an active learning method to learn the phase transition boundary in these models. We used random forest classifiers, but any other classifier can be used. We have also tried using support vector machines with linear kernels, and the results are similar.

There are multiple uses for this kind of analysis. Modeling the response surface puts the focus on the parameters instead of the features themselves. For example, a regression analysis might show that the Mile1 feature is highly significant for prediction solar panel adoption. However, analyzing the response surface might show that increasing the coefficient for Net Present Value would also be a way to cross the phase transition boundary and increase the number of adopters. The first insight (about Mile1) suggests that one way to increase adoption is to give away solar panels to some households in such a way that other households also start adopting. This is a version of the influence maximization problem [12] and has been studied in this domain [10]. However, this might be quite expensive. The second insight suggests that another way to increase adoption might be to make people more aware of the Net Present Value to them, thereby increasing the weight they attach to it. This informational campaign would be much cheaper.

Comparing two regions, even if models are made by different researchers with different data sources and assumptions about model structure can be very instructive. It can help to answer the question of how likely is the observed difference between the two regions. This can offer fundamental insight into whether different policy approaches are needed for different regions.

There are multiple avenues for further research. The robustness of the method needs further study. In the case of contagion models, since we are able to exploit the property that model variance increases sharply at the phase transition boundary, we can find actual boundary points. This allows us to start the active learning with very few points but avoid the problems associated with limited data. If the boundary is not so well-defined, we might need to do more simulation runs, even with active learning, to characterize the regions properly. In general, we don't have a way of determining how many points are needed to learn the boundaries between regions. An important direction of research is to determine that, or at least come up with an explainable heuristic.

Another important direction of research is to ask, what are the changes necessary to minimize the characteristic distance or disagreement between two models? We might wish to come up with succinct explanations for the reasons two models disagree. As agent-based simulations are getting larger and more complex, this kind of explainability is becoming increasingly important. Hopefully, the present work will motivate further work along these lines.

ACKNOWLEDGMENTS

This work was supported in part by DOE grant DE-EE0007660, NIH grant 1R01GM109718, NSF CRISP 2.0 grant 1832587, and DTRA CNIMS contract HDTRA1-11-D-0016-0001. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

REFERENCES

[1] Abhijin Adiga, Chris Barrett, Stephen Eubank, Chris J. Kuhlman, Madhav V. Marathe, Henning Mortveit, S. S. Ravi, Daniel J. Rosenkrantz, Richard E. Stearns,

Samarth Swarup, and Anil Vullikanti. 2019. Validating agent-based models of large networked systems. In *Winter Simulation Conference (WSC)*.

[2] Robert Axtell, Robert Axelrod, Joshua M. Epstein, and Michael D. Cohen. 1996. Aligning Simulation Models: A Case Study and Results. *Computational and Mathematical Organization Theory* 1, 2 (1996), 123–141.

[3] Russell R. Barton and Martin Meckesheimer. 2006. Metamodel-Based Simulation Optimization. In *Simulation*, Shane G. Henderson and Barry L. Nelson (Eds.). Handbooks in Operations Research and Management Science, Vol. 13. Elsevier, 535 – 574.

[4] Gnana K. Bharathy and Barry Silverman. 2010. Validating Agent Based Social Systems Models. In *Proceedings of the Winter Simulation Conference (WSC '10)*. Winter Simulation Conference, 441–453. http://dl.acm.org/citation.cfm?id=2433508.2433559

[5] G. E. P. Box and K. B. Wilson. 1951. On the Experimental Attainment of Optimum Conditions. *Journal of the Royal Statistical Society, Series B (Methodological)* 13, 1 (1951), 1–45.

[6] Richard M. Burton and Børge Obel. 1999. The Challenge of Validation and Docking. In *Proceedings of the Workshop on Agent Simulation: Applications, Models, and Tools*. Argonne National Laboratory, 216–221.

[7] Kathleen M. Carley, Natalia Y. Kamneva, and Jeff Reminga. 2004. *Response Surface Methodology*. CASOS Technical Report CMU-ISRI-04-136. Carnegie Mellon University.

[8] Nick Collier. 2003. Repast: An extensible framework for agent simulation. *The University of Chicago's Social Science Research* 36 (2003), 2003.

[9] Arindam Fadikar, Dave Higdon, Jiangzhuo Chen, Bryan L. Lewis, Srini Venkatramanan, and Madhav V. Marathe. 2018. Calibrating a Stochastic Agent-based Model using Quantile-based Emulation. *SIAM/ASA J. Uncertainty Quantification* 6, 4 (2018), 1685–1706.

[10] Aparna Gupta, Samarth Swarup, Achla Marathe, Anil Vullikanti, Kiran Lakkaraju, and Joshua Letchford. 2018. Designing Incentives to Maximize the Adoption of Rooftop Solar Technology (Extended Abstract). In *Proceedings of the 17th International Conference on Autonomous Agents and Multi-Agents Systems (AAMAS)*. Stockholm, Sweden.

[11] Zhihao Hu, Xinwei Deng, Achla Marathe, Samarth Swarup, and Anil Vullikanti. 2019. Decision-Adjusted Modeling for Imbalanced Classification: Predicting Rooftop Solar Panel Adoption in Rural Virginia. In *Proceedings of The Computational Social Science Conference*.

[12] David Kempe, Jon Kleinberg, and Éva Tardos. 2003. Maximizing the Spread of Influence through a Social Network. In *Proceedings of KDD*. Washington, DC, USA.

[13] Francesco Lamperti, Andrea Roventini, and Amir Sani. 2018. Agent-based Model Calibration Using Machine Learning Surrogates. *Journal of Economic Dynamic & Control* 90 (2018), 366–389.

[14] David D. Lewis and William A. Gale. 1994. A Sequential Algorithm for Training Text Classifiers. In *SIGIR '94*, Bruce W. Croft and C. J. van Rijsbergen (Eds.). Springer London, London, 3–12.

[15] Marcus A Louie, Kathleen M Carley, Laleh Haghshenass, John C Kunz, Raymond E Levitt, et al. 2003. Model comparisons: docking ORGAHEAD and SimVision. In *Proceedings of NAACSOS conference, Pittsburgh, PA*. Citeseer.

[16] H. Mao, F. Guo, X. Deng, and Z. Doerzaph. 2019. Decision-adjusted Driver Risk Predictive Model using Kinematics Information. *submitted to IEEE Transactions on Intelligent Transportation Systems* (2019).

[17] Nelson Minar, Roger Burkhart, Chris Langton, Manor Askenazi, et al. 1996. The swarm simulation system: A toolkit for building multi-agent simulations. (1996).

[18] H.S. Mortveit and C.M. Reidys. 2007. *An Introduction to Sequential Dynamical Systems*. Springer Verlag.

[19] H. Gonda Neddermeijer, Gerrit J. van Oortmarssen, Nanda Piersma, and Rommert Dekker. 2000. A Framework for Response Surface Methodology for Simulation Optimization. In *Proceedings of the 32nd Conference on Winter Simulation (WSC '00)*, J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick (Eds.). Society for Computer Simulation International, San Diego, CA, USA, 129–136. http://dl.acm.org/citation.cfm?id=510378.510401

[20] Michael North and Charles M Macal. 2002. The beer dock: Three and a half implementations of the beer distribution game. *SwarmFest, Swarm Development Group* (2002).

[21] Victor M. Pérez, John E. Renaud, and Layne T. Watson. 2002. Adaptive Experimental Design for Construction of Response Surface Approximations. *AIAA Journal* 40, 12 (2002), 2495–2503.

[22] Warren Volk-Makarewicz and Catherine Cleophas. 2017. A Meta-algorithm for Validating Agent-based Simulation Models to Support Decision Making. In *Proceedings of the 2017 Winter Simulation Conference (WSC '17)*. IEEE Press, Piscataway, NJ, USA, Article 102, 12 pages. http://dl.acm.org/citation.cfm?id=3242181.3242289

[23] Jin Xu, Yongqin Gao, and Gregory Madey. 2003. A docking experiment: Swarm and repast for social network modeling. In *Seventh Annual Swarm Researchers Meeting (Swarm2003)*. 1–9.

[24] Haifeng Zhang, Yevgeniy Vorobeychik, Joshua Letchford, and Kiran Lakkaraju. 2016. Data-driven agent-based modeling, with application to rooftop solar adoption. *Auton Agent Multi-Agent Syst* 30, 6 (2016), 1023–1049. https://doi.org/10.1007/s10458-016-9326-8