Trusted Machine Learning for Probabilistic Models

Shalini Ghosh, Patrick Lincoln, Ashish Tiwari SHALINI,LINCOLN,TIWARI@CSL.SRI.COM
Computer Science Laboratory, SRI International

Xiaojin Zhu JERRYZHU@CS.WISC.EDU
Department of Computer Sciences, University of Wisconsin-Madison

Abstract

In several mission-critical domains (e.g., self-driving cars, cybersecurity, robotics) where machine learning algorithms are being used heavily, it is becoming increasingly important to ensure that the learned models satisfy some domain properties (e.g., temporal constraints). Towards this goal, we propose *Trusted Machine Learning (TML)*, wherein we combine the strengths of machine learning and model checking. If the desired logical properties are not satisfied by a trained model, we modify either the model ('model repair') or the data from which the model is learned ('data repair'). We outline a concrete case study based on the Markov Chain model of a car controller for 'lane changing' — we demonstrate how we can ensure that such a model, learned from data, satisfies properties specified in Probabilistic Computation Tree Logic (PCTL).

1. Introduction

In machine learning (ML), a model is typically trained on training data to be able to generalize better on unseen test data — this usually involves learning some parameters of the model by optimizing an objective function (e.g., likelihood of observing the training data given the model). Additionally, we often want the model to satisfy certain constraints. In the ML literature, there is a rich history of learning under constraints (Dietterich, 1985; Miller & MacKay, 1994). Different types of constrained learning algorithms have been proposed for various kinds of constraints and algoritms. Propositional constraints on size (Bar-Hillel et al., 2005), monotonicity (Kotlowski & Slowiński, 2009), time and ordering (Laxton et al., 2007), etc. have been incorporated into learning algorithms using techniques like constrained optimization (Bertsekas, 1996) and constraint programming (Raedt et al., 2010). First order logic constraints have also been introduced into ML models (Mei et al., 2014; Richardson & Domingos, 2006). Some problems have constraints that are better defined over temporal sequences or trajectories in the model — these constraints can be succintly represented in temporal logic, e.g., Probabilistic Computation Tree Logic (PCTL)(Sen et al., 2006). To this end, we develop a methodology to train probabilistic ML models that satisfy properties specified in temporal logic, called *Trusted Machine Learning* (TML).

We illustrate the need for TML by a case-study where we consider training an automatic car controller for the situation where there is a slow-moving car in front. We provide training data (D) in the form of example traces of driving in this situation in a simulated environment, and model the controller (M), using a Discrete-Time Markov Chain (DTMC)(Sen et al., 2006). In our application, we want the car controller to respect certain safety properties. For example, we want the controller to ensure that it causes the car to either change lanes or reduce speed with very high probability. This can be stated as a PCTL property $\Pr_{>0.99}[F(changedLane \mid reducedSpeed)]$, where F is the *eventually* operator in PCTL logic. Such temporal logic properties are useful for specifying important characteristics of mission-critical domains (e.g., self-driving cars, cybersecurity). If the trained probabilistic ML model does not satisfy a desired temporal logic property, we propose two approaches to ensure that the model satisfies the property:

1) *Model repair*: The model can be changed "locally" (e.g., in a probabilistic model we can suitably modify transition probabilities along the path of the unsafe trajectory), so that the "repaired" model satisfies the desired property.

2) *Data repair*: The training data can be modified suitably (e.g., modifying data features, or changing noisy target labels), so that the model trained on this "repaired" data satisfies the desired property.

Here are the main contributions of our work:

1) We formulate the problem of Trusted Machine Learning (TML), where we want a ML model trained on data to sat-

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

isfy properties specified in temporal logic.

2) We propose two approaches to solve TML, Model Repair and Data Repair, which apply principles of *parametric model checking* and *machine teaching* respectively. For probabilistic ML models that have to satisfy temporal logic properties, we show how Model Repair can be solved using non-linear optimization, and how the Data Repair problem can be solved using the Model Repair formulation under certain assumptions.

3) We present a TML case-study for Model Repair on a DTMC car controller model and PCTL properties, using the PRISM model checker and AMPL non-linear solver.

2. Motivating Example

Let us consider the example of training a ML model as an autonomous car controller — the underlying model we want to learn is a Discrete-Time Markov Chain (DTMC), which is defined as a tuple $M = (S, s_0, P, L)$ where S is a finite set of states, $s_0 \in S$ is the initial state, $P : S \times S \rightarrow [0, 1]$ is a function such that $\forall s \in S, \sum_{s' \in S} \Pr(s, s') = 1$, and $L : S \rightarrow 2^{AP}$ is a labeling function assigning labels to states, where AP is a set of atomic propositions. Figure 1 shows the Markov Chain of an autonomous car controller that determines the action of the controller when confronted with a slow-moving car in front, while Table 1 describes the semantics of the different states and labels. Note that the figure has parameters p and q in the state transition probabilities, which are used for model repair (details in Section 3.1).

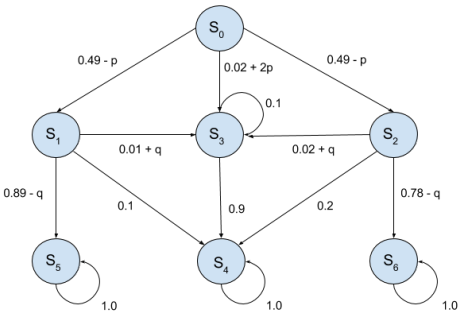

Figure 1. Markov Chain of car controller for changing lanes.

Let us consider that we have some property ϕ in PCTL that we want the model M to satisfy. The PCTL logic is defined over the DTMC model, where properties are specified as $\phi = \Pr_{\sim b}(\psi)$, with $\sim \in \{<, \leq, >, \geq\}$, $0 \leq b \leq 1$, and ψ a path formula defined using the X (next) and $\cup^{\leq h}$ (bounded/unbounded until) operators, for integer h. A state s of M satisfies $\phi = \Pr_{\sim b}(\psi)$, denoted as $M, s \models \phi$, if $\Pr(Path_M(s, \psi)) \sim b$; i.e., the probability of taking a path in M starting from s that satisfies ψ is $\sim b$, where path is defined as a sequence of states in the model M. PCTL also uses the *eventually* operator F defined as $F\phi = true \cup \phi$, i.e., ϕ is eventually true. For example, let us consider a DTMC M in Figure 1 with start state S_0. The "reducedSpeed" predicate is true in the state S_4, while "changedLane" predicate is true in states S_5 and S_6. Now, a probabilistic safety property can be specified as: $\phi = \Pr_{>0.99}[F(changedLane \mid reducedSpeed)]$. The model M will satisfy the property ϕ only if any path starting from S_0 eventually reaches S_4, S_5 or S_6 with probability > 0.99.

Table 1. States and labels of Car Controller DTMC.

S	Description	Labels
0	Initial state	keepSpeed, keepLane
1	Moving to left lane	keepSpeed, changingLane
2	Moving to right lane	keepSpeed, changingLane
3	Remain in same lane with same speed	keepSpeed, keepLane
4	Remain in same lane with reduced speed	reducedSpeed, keepLane
5	Moved to left lane	changedLane
6	Moved to right lane	changedLane

3. Approach

In TML we focus on *efficient* mechanisms of making an existing ML model M satisfy property ϕ with minimal changes to M — so we don't consider approaches like creating product models using M and ϕ (Sadigh et al., 2014b), since those typically lead to a large (often exponential) blowup in the state space of the resulting model (Kupferman & Vardit, 1998). Figure 2 shows the details of the TML flow. Let us consider that we train the model M from a given training data D. We first use model checking on M to see if M satisfies ϕ — if so, we output M. Otherwise, if M does not satisfy ϕ, we next try to do a "local" modification of model M using Model Repair (details in Section 3.1). If that becomes infeasible, we finally try to do Data Repair of the data D (details in Section 3.2).

3.1. Model Repair

We first learn the model M from data D using standard ML techniques, e.g., maximum likelihood. When M does not satisfy the given logical property ϕ, Model Repair tries to minimally perturb M to M' to satisfy ϕ. The Model Repair problem for probabilistic systems can be stated as follows:

Given a probabilistic model M and a probabilistic temporal logic formula ϕ, if M fails to satisfy ϕ, can we find a variant M' that satisfies ϕ such that the cost associated in modifying the transition flows of M to obtain M' is minimized, where M' differs from M only in transition flows related to controllable states in M?

We will illustrate the methodology of Model Repair using the DTMC in Section 2. A DTMC M with n states can be turned into a *controllable* parametric DTMC by considering an $n \times n$ matrix Z that specifies which states of M are controllable. The matrix Z gives a mechanism for altering or controlling the behavior of M for repair — Z is added to the transition matrix P of the DTMC, i.e., $P' = P + Z$. A state of the DTMC is controllable if at least one transition in/out of that node has a non-zero Z value. Figure 1 shows the parametric DTMC, where some states are controllable but others are not. Note that there are certain constraints on Z in a controllable DTMC to make the Model Repair approach feasible, namely $\forall s \sum_{t \in S} Z(s,t) = 0$ (Bartocci et al., 2011).

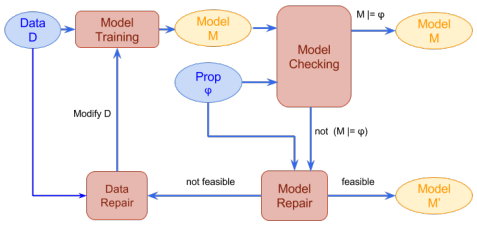

Figure 2. Overall flow of TML.

Let us consider the non-zero values in Z to be the vector of variables $\mathbf{v} = v_1 \ldots v_k$. Finding the optimal Z (that causes the minimal change to M) can be cast as the following optimization problem:

$$\arg \min_{\mathbf{v}} g(\mathbf{v}), \tag{1}$$

$$M', S_0 \models \phi, \tag{2}$$

$$P'(i,j) = 0 \text{ iff } P(i,j) = 0, 1 \leq i,j \leq |S|. \tag{3}$$

In Equation 1, $g(\mathbf{v})$ is a cost function that encodes the cost of making the perturbation to model parameters — a typical function is the sum of squares of the perturbation variables, i.e., $g(\mathbf{v}) = ||\mathbf{v}||^2 = v_1^2 + \ldots + v_n^2$. Equation 2 checks if the modified model M' (with S_0 as its initial state) satisfies property ϕ. Equation 3 forces that no transitions are added or dropped in M' w.r.t. M, only the transition probabilities are modified. This condition encodes the "local" nature of the change of M, since Z cannot change the structure or stochasticity of the underlying DTMC model — this keeps the model repair problem tractable.

Model Repair can be reduced to a nonlinear optimization problem with a minimal-cost objective function using parametric model checking (Bartocci et al., 2011), yielding an efficient solution technique. Using *parametric model checking* (Hahn et al., 2010), Equations 1-3 become:

$$\min \quad g(\mathbf{v}), \tag{4}$$

$$f(\mathbf{v}) \sim b, \tag{5}$$

$$\forall v \in \mathbf{v} : 0 < y(v) < 1. \tag{6}$$

where $\forall v_k \in \mathbf{v}, y(v_k) = P(i,j) + v_k$, where (i,j) are the indices in the Z matrix corresponding to the non-zero value v_k. This reparameterization of Equation 2, encoding the satisfiability of ϕ in M, to the non-linear equation $f(v)$ in Equation 5 is obtained using a parametric model checker, e.g., PRISM (Kwiatkowska et al., 2011). Solving the non-linear objective function in Equation 4 with the non-linear constraints in Equation 5-6 would give us the optimal Z that transforms M to M' — we can do that using a non-linear optimization tool, e.g., AMPL (Fourer et al., 1989). If the nonlinear optimization problem has a feasible solution, it gives us the optimal values of Z that makes the resulting model M' satisfy ϕ.

3.2. Data Repair

The non-linear optimization problem for Model Repair does not always have a feasible solution (Bartocci et al., 2011). If Model Repair of M becomes infeasible, we take the approach of modifying dataset D to D' so that the model trained on D' satisfies ϕ — we solve this using the Data Repair approach, which is a variant of *machine teaching* (Zhu, 2015) and is defined as:

Given a logical property ϕ that we would like to satisfy and a probabilistic ML model M parameterized by Θ that we want to train using a dataset D, if M does not satisfy ϕ, can we "perturb" the given data D by the minimal amount to get D' such that a new model M' trained on D' satisfies the property ϕ?

Based on the machine teaching formulation (Mei & Zhu, 2015), we define Data Repair as:

$$\arg \min_{D', \Theta^*} E_T(D, D'), s.t. \tag{7}$$

$$M_{\Theta^*}, S_0 \models \phi \tag{8}$$

$$\Theta^* \in \arg\min_{\Theta} [R_L(D', \Theta) + \lambda \Omega(\Theta)], \tag{9}$$

$$s.t., g(\Theta) \leq 0, h(\Theta) = 0. \tag{10}$$

Here, the inner optimization models the standard machine learning objective of regularized empirical risk minimization, consisting of the empirical risk function R_L and the regularizer Ω. E_T is the teaching "effort" function of modifying the given dataset D to D', M_{Θ^*} indicates a model that is parameterized by Θ^*, while g and h are other domain constraints. Let us consider that the dataset D is transformed to D' using a data perturbation vector p. For example, let us consider that a small set of data points need to be dropped from D for the resulting trained model to satisfy ϕ (e.g., those points could have noisy features or labels). So, each datapoint d_i in D is multiplied by $(1 - p_i)$, where $p_i = 1$ indicates that the point is dropped — in this case, $p = \{p_1 \ldots p_n\}$, where $n = |D|$. Also, let us consider that the effort function is characterized by the magnitude of the data perturbation, i.e., $E_T(D, D') = ||p||^2$. Using these transforms, Equations 7-10 can be reformulated as:

$$\arg \quad \min_{p,\Theta^*} ||p||^2, s.t. \tag{11}$$

$$M_{\Theta^*}, S_0 \models \phi, \tag{12}$$

$$\Theta^* \in \arg\min_{\Theta}[R'_L(D, p, \Theta) + \lambda\Omega(\Theta)], \tag{13}$$

$$s.t., g(\Theta) \leq 0, h(\Theta) = 0. \tag{14}$$

Note that $R'_L(D, p, \Theta)$ is a reparameterization of $R_L(D', \Theta)$, where we use the fact that D' is obtained by perturbing D using p. To solve the non-linear optimization formulation in Equations 11-14, we first solve the inner optimization in Equations 13-14 using maximum likelihood — this gives us a DTMC model $M(p)$, where the transition probabilities are *rational functions* of the data perturbation vector p. The outer optimization in Equations 11-12 can then be reformulated as:

$$\arg \quad \min_{p} ||p||^2, s.t. \tag{15}$$

$$M_p, S_0 \models \phi. \tag{16}$$

This is now similar to the Model Repair formulation in Equations 1-2, which can be solved using parametric model checking and non-linear optimization as discussed in Section 3.1. In this case, we are considering data points being removed — we can come up with similar formulations for Data Repair when we consider data points being added or replaced. Note that the Data Repair approach is useful when Model Repair becomes infeasible, a motivation for which is shown through our case study example in Section 4.

4. Case Study: Car Controller

In this case-study, we will use the running example of the car-controller described in Section 2. We assume that the initial values of the DTMC model in Figure 1 are learned using maximum likelihood estimation from simulated car traces. We work through 3 different cases:

Model satisfies property: Let us consider the property $\texttt{Prop1} = \Pr_{>0.99}[F(changedLane|reduceSpeed)]$ (as described in Section 1). We consider a model M in PRISM corresponding to Figure 1 where $p = q = 0$, i.e., we don't consider any Model Repair. When we run PRISM model checker on this model for `Prop1`, PRISM reports that the property is "true", i.e., the property is always satisfied from the initial state S_0. This indicates that the model satisfies property `Prop1`, without the need for any modifications.

Model Repair gives feasible solution: Let us consider the property $\texttt{Prop2} = \Pr_{>0.8}[F(reduceSpeed)]$. When we run PRISM model checking on M with $p = q = 0$ for `Prop2`, PRISM reports that the property is "false", i.e., the property is not satisfied from the initial state S_0. We subsequently run parametric model checking on M, which converts `Prop2` to a non-linear parametric equation: $0.8 < 0.0001(-20000qp + 16700p + 9800q + 1817)$. Plugging this as a constraint into AMPL, we minimize the objective function for model repair: $p^2 + q^2$, also considering other constraints that ensure that the repaired transition probabilities lie in $[0, 1]$. AMPL gives the solution $p = 0.3609, q = 0.0601$, which are the values of p and q by which model M needs to be perturbed to satisfy `Prop2`.

Model Repair gives infeasible solution: Let us consider the property $\texttt{Prop3} = \Pr_{<0.1}[F(reduceSpeed)]$. This property is not satisfied by M — parametric model checking and non-linear optimization states this to be a "infeasible problem", which indicates that Model Repair cannot perturb M in order to satisfy `Prop3`. [1] In this case we would need some method other than Model Repair to make M satisfy `Prop3`, e.g., Data Repair.

5. Related Work

There has been work on training models that capture dynamical/temporal behavior, e.g., DBNs (Neapolitan, 2003), LSTMs (Hochreiter & Schmidhuber, 1997). There have also been efforts in learning temporal logic relations (Maggi et al., 2013). However, to the best of our knowledge, temporal logic constraints have not been incorporated into ML models before. (Sadigh et al., 2014a) study the problem of human driver behavior using Convex Markov Chains, and show how we can verify PCTL properties for these models. (Puggelli et al., 2013) show how Convex MDPs can be modified to satisfy PCTL formulas. However, these methods follow techniques different from Model and Data Repair.

6. Conclusions and Future Work

Our approach to Trusted Machine Learning (TML) uses principles from Formal Methods (specifically model checking) for learning ML models that satisfy properties in temporal logic. Our key insight in TML is using techniques like Model Repair and Data Repair that "locally repair" the model or the data respectively, to satisfy the properties. In this paper, we have developed the TML control flow for Markov Chain models satisfying PCTL properties, and discussed a possible application of our approach to the domain of car controllers.

In the future, we would like to characterize the different types of Data Repair in TML, and extend the car controller case study to include Data Repair. We would also like to extend TML to other probabilistic models (e.g., Markov Decision Processes) and other types of logical properties (e.g., LTL). We would like to apply TML to other misson-critical domains, e.g., robot control, cyber security. Another challenge is to adapt TML to non-probabilistic models (e.g., SVM, regression models). We would also like to explore connections between TML and probabilistic CEGAR algorithms (Hermanns et al., 2008).

[1] Details at: http://www.csl.sri.com/users/shalini/tml

References

Bar-Hillel, Aharon, Hertz, Tomer, Shental, Noam, and Weinshall, Daphna. Learning a Mahalanobis metric from equivalence constraints. *JMLR*, 6, 2005.

Bartocci, Ezio, Grosu, Radu, Katsaros, Panagiotis, Ramakrishnan, C. R., and Smolka, Scott A. Model repair for probabilistic systems. In *Tools and Algorithms for the Construction and Analysis of Systems*, 2011.

Bertsekas, Dimitri P. *Constrained Optimization and Lagrange Multiplier Methods (Optimization and Neural Computation Series)*. Athena Scientific, 1996.

Dietterich, Thomas Glen. *Constraint Propagation Techniques for Theory-driven Data Interpretation (Artificial Intelligence, Machine Learning)*. PhD thesis, 1985.

Fourer, R., Gay, D. M., and Kernighan, B. Algorithms and model formulations in mathematical programming. chapter AMPL: A Mathematical Programming Language. 1989.

Hahn, Ernst Moritz, Hermanns, Holger, Wachter, Björn, and Zhang, Lijun. PARAM: A model checker for parametric Markov models. In *CAV*, 2010.

Hermanns, Holger, Wachter, Björn, and Zhang, Lijun. In *CAV*, 2008.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural Comput.*, 9(8), 1997.

Kotlowski, Wojciech and Slowiński, Roman. Rule learning with monotonicity constraints. In *ICML*, 2009.

Kupferman, O. and Vardit, M. Y. Freedom, weakness, and determinism: from linear-time to branching-time. In *Proc. IEEE Symposium on Logic in Computer Science*, 1998.

Kwiatkowska, M., Norman, G., and Parker, D. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, 2011.

Laxton, B., Lim, J., and Kriegman, D. Leveraging temporal, contextual and ordering constraints for recognizing complex activities in video. In *CVPR*, 2007.

Maggi, Fabrizio Maria, Burattin, Andrea, Cimitile, Marta, and Sperduti, Alessandro. Online process discovery to detect concept drifts in LTL-based declarative process models. In *On the Move to Meaningful Internet Systems: OTM Conf.*, 2013.

Mei, Shike and Zhu, Xiaojin. Using machine teaching to identify optimal training-set attacks on machine learners. In *AAAI*, 2015.

Mei, Shike, Zhu, Jun, and Zhu, Jerry. Robust RegBayes: Selectively incorporating first-order logic domain knowledge into Bayesian models. In *ICML*, 2014.

Miller, K. D. and MacKay, D. J. C. The role of constraints in Hebbian learning. *Neural Computation*, 6(1), 1994.

Neapolitan, Richard E. *Learning Bayesian Networks*. Prentice-Hall, Inc., 2003.

Puggelli, Alberto, Li, Wenchao, Sangiovanni-Vincentelli, Alberto, and Seshia, Sanjit A. Polynomial-time verification of PCTL properties of MDPs with convex uncertainties. In *CAV*, 2013.

Raedt, Luc De, Guns, Tias, and Nijssen, Siegfried. Constraint programming for data mining and machine learning. In *AAAI*, 2010.

Richardson, Matthew and Domingos, Pedro. Markov logic networks. In *Machine Learning*, 2006.

Sadigh, Dorsa, Driggs-Campbell, Katherine, Puggelli, Alberto, Li, Wenchao, Shia, Victor, Bajcsy, Ruzena, Sangiovanni-Vincentelli, Alberto L., Sastry, S. Shankar, and Seshia, Sanjit A. Data-driven probabilistic modeling and verification of human driver behavior. In *Formal Verification and Modeling in Human-Machine Systems, AAAI Spring Symposium*, 2014a.

Sadigh, Dorsa, Kim, Eric S., Coogan, Samuel, Sastry, S. Shankar, and Seshia, Sanjit A. A learning based approach to control synthesis of Markov decision processes for linear temporal logic specifications. In *53rd IEEE Conf. on Decision and Control (CDC)*, 2014b.

Sen, Koushik, Viswanathan, Mahesh, and Agha, Gul. *Tools and Algorithms for the Construction and Analysis of Systems*, chapter Model-Checking Markov Chains in the Presence of Uncertainties. 2006.

Zhu, Xiaojin. Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *AAAI*, 2015.