LGML: Logic Guided Machine Learning

Joseph Scott, Maysum Panju, Vijay Ganesh
University of Waterloo
200 University Ave West, Waterloo, Ontario, Canada N2L 3G1, +1-519-888-4567
{joseph.scott, mhpanju, vijay.ganesh } @uwaterloo.ca

Abstract

We introduce *Logic Guided Machine Learning* (LGML), a novel approach that symbiotically combines machine learning (ML) and logic solvers to learn mathematical functions from data. LGML consists of two phases, namely a *learning-phase* and a *logic-phase* with a corrective feedback loop, such that, the learning-phase learns symbolic expressions from input data, and the logic-phase cross verifies the consistency of the learned expression with known auxiliary truths. If inconsistent, the logic-phase feeds back "counterexamples" to the learning-phase. This process is repeated until the learned expression is consistent with auxiliary truth. Using LGML, we were able to learn expressions that correspond to the Pythagorean theorem and the sine function, with several orders of magnitude improvements in data efficiency compared to an approach based on an out-of-the-box multi-layered perceptron (MLP).

Logic Guided Machine Learning (LGML) is a new approach that combines an ML model (the learning-phase) with a logic solver (the logic-phase) in a corrective feedback loop from the logic-phase to the learning-phase (see Figure 1). LGML takes the following as input: labeled data corresponding to an unknown target mathematical function f to be learned and an auxiliary truth ψ (theorems or invariants over f in a suitable fragment of mathematics). The learning-phase fits a symbolic function $\hat{f}$, that approximates f, over this data which is then fed to a logic solver (e.g., SAT/SMT solvers) along with the auxiliary truth ψ. If the symbolic expression $\hat{f}$ is inconsistent with the given auxiliary truth ψ, the logic-phase feeds back the data point that violates the auxiliary truth the "strongest". While the ground truth function f underlying the dataset is unknown to the system, LGML does require access to an oracle to produce labels to the returned point from the logic-phase. This process is repeated until $\hat{f}$ and ψ are consistent with each other, i.e., $\hat{f} \models \psi$ (By this notation, we mean $\forall x.\psi(x, \hat{f}(x))$ is true). This work is inspired by (Clarke et al. 2000).

Input to LGML: The input to LGML is a labeled dataset that relates the inputs and output of some function f whose symbolic representation is not known to the system, and an
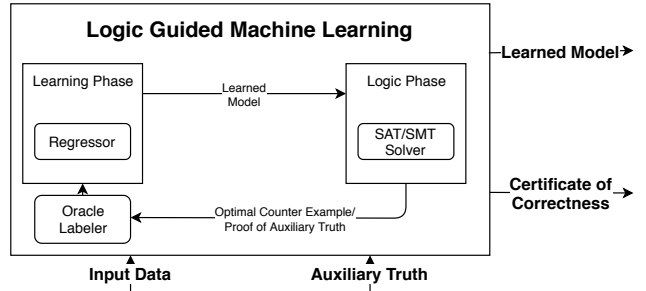
Figure 1: Architecture Diagram of of LGML

auxiliary theorem or truth ψ known to be true for the inputs and output f. For example, consider $f(a, b)$ to be the function that takes as input the length a and b of the sides of a right-angled triangle and computes the length of its hypotenuse using the Pythagorean theorem. Then, one possible ψ is the triangle inequality $a + b > f(a, b)$. Even if one may not quite know the statement of the Pythagorean theorem, one may have a large sample of right-angled triangles (input data set) and know the statement of the triangle inequality (auxiliary truth). Combinatorial mathematicians and physicists routinely have access to such data sets and auxiliary truths and want to learn some previously unknown function or invariant over such data (Larson and Van Cleemput 2017).

Output of LGML: LGML outputs some function $\hat{f}$ that fits the labeled dataset and is consistent with ψ. Unlike classical regressors, LGML starts with a small dataset and augments it with additional points during runtime, obtained via a logical-phase.

The learning-phase of LGML: The learning-phase applies a regression algorithm on the current dataset to obtain a math expression $\hat{f}$ that fits the input data. Examples of regression algorithms include symbolic regression methods and deep neural networks. Importantly, we require the regressor to fit all data points with near-perfect accuracy.

The logic-phase of LGML: The logic-phase checks whether the learned function $\hat{f}$ is consistent with the input auxiliary truth ψ, denoted as $\hat{f} \models \psi$. As long as this is not
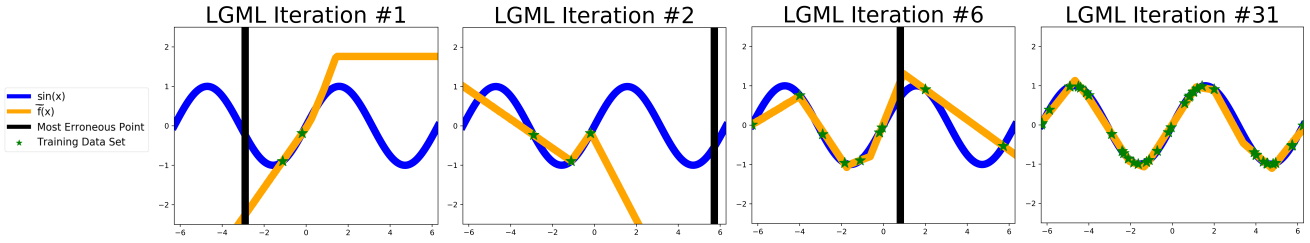
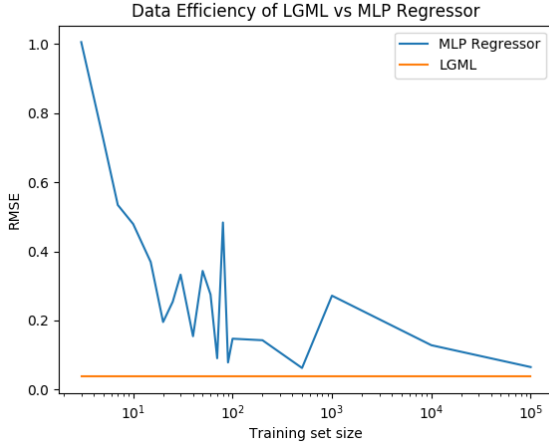Figure 2: Visualization of LGML for select iterations

Figure 3: RMSE of LGML vs. standard MLP regressor.

the case, LGML computes the 'strongest' counterexample resulting in inconsistency, inserts it into the training set, and repeats the learning-phase on the augmented training data. First note that ψ is a logical equation in terms of the feature space and the unknown function f, or more precisely, in terms of the output of $\hat{f}$ that approximates f. Hence, in its logic-phase, LGML constructs a satisfiability query that checks whether $\hat{f} \models \psi$. If inconsistent, a counterexample is extracted from the logic solver. LGML finds the 'strongest' such counterexample and returns it to the learning-phase.

Auxiliary Truth as a Satisfiability Query: We weaken the auxiliary truth ψ (a quantifier-free mathematical formula) to be written with an error term ϵ, denoted as ψ_ϵ. Suppose the auxiliary truth ψ is of the form $\alpha = \beta$, where α and β are well-formed symbolic expressions. Then we construct the query:

$$\neg\psi_\epsilon := |\alpha[f/\hat{f}] - \beta[f/\hat{f}]| > \epsilon$$

Further, if ψ is of the form of an inequality, (e.g $\alpha > \beta$), then:

$$\neg\psi_\epsilon := \beta[f/\hat{f}] - \alpha[f/\hat{f}] > \epsilon$$

where the $/$ is the logical substitution. The satisfiability is checked via a floating-point (FP) SMT solver for a fixed ϵ. We model $\hat{f}$ with 16-bit precision and use CVC4 as the backend FP SMT solver.

Extracting Proof and the Strongest Erroneous Point: We use a variant of the well-known bisection method to find both proofs and the strongest erroneous points with respect to an auxiliary truth. By making multiple queries to an FP SMT solver, we compute ϵ^* such that

$$\hat{f} \models \psi_\epsilon$$

for all $\epsilon > \epsilon^*$, and for all $\epsilon < \epsilon^*$,

$$\hat{f} \not\models \psi_\epsilon$$

For a target machine error ρ, we first query $\epsilon = \rho$. The terminating condition of LGML is the proof of $\hat{f} \models \psi_\rho$. We exponentially increase ϵ by doubling its value until a SAT result (i.e $\hat{f} \not\models \psi_\epsilon$). An UNSAT result (i.e $\hat{f} \models \psi_\epsilon$) establishes an interval containing ϵ^*, which we narrow using the bisection method until convergence on ϵ^* and as consequently the strongest erroneous point.

Evaluation: We empirically tested LGML on two tasks: learning the function f corresponding to Pythagorean theorem and the sine function. For brevity, we focus on sine function here. The auxiliary truth we use is $\psi := \sin^2(x) + \cos^2(x) = 1$, and the satisfiability query that we use is $|\hat{f}(x)^2 + \left(\frac{d}{dx}\hat{f}(x)\right)^2 - 1| > \epsilon$, for various ϵ as described until ϵ^* is found. As the base learning model, we use an MLP regressor with two hidden layers of 3 nodes each. Figure 2 visualizes select iterations of LGML. As can be seen, the LGML learns the Sine function almost perfectly.

We evaluate LGML at the 30th iteration (for a total of 32 training points) with a testing set of 10,000 points and compute an RMSE of 0.037. As a baseline, an MLP regressor was trained to learn the same function without using LGML, but given increasingly large training sets. Figure 3 demonstrates that the LGML model achieved lower error scores on learning $f(x) = \sin(x)$ using just 32 training points than the non-LGML MLP model when given even 100,000 points.

References

Clarke, E.; Grumberg, O.; Jha, S.; Lu, Y.; and Veith, H. 2000. Counterexample-guided abstraction refinement. In *International Conference on Computer Aided Verification*, 154–169. Springer.

Larson, C. E., and Van Cleemput, N. 2017. Automated conjecturing iii. *Annals of Mathematics and Artificial Intelligence* 81(3-4):315–327.