Debugging the Machine Learning Pipeline

Jerry Zhu

University of Wisconsin-Madison

joint work with Xuezhou Zhang, Stephen Wright
Interpretable ML Symposium, NIPS 2017
Debugging provides an opportunity for machine learning interpretability.
Harry Potter toy example

![Graph showing education levels for Hermione and Goyle. Hermione is placed at the 1 education level on the muggle-born side, and Goyle is placed at the 0 education level on the pure-blood side.](image)
Hired by the Ministry of Magic?

+ yes
○ no
Data contain historical biases

Learned vs. ideal decision boundary

(RBF kernel logistic regression)
Trusted items

- obtained by expensive vetting
- insufficient to learn from

![Graph showing the relationship between magical heritage and education. The x-axis represents magical heritage ranging from 0 to 1, and the y-axis represents education ranging from 0 to 1. There are two points marked on the graph: one at (0.5, 0.5) and another at (1, 1). The line connecting these points suggests a direct relationship between the two variables.]
Debugging using trusted items

- propose training label bugs
- flipping them makes re-trained model agree with trusted items

![Graph showing magical heritage vs. education with data points and a decision boundary.](image)
Proposed bugs

- given to experts to interpret

![Graph showing data points on magical heritage vs. education.](image)
The ML pipeline

\[
\text{data } (X, Y) \rightarrow \text{learner } \ell \rightarrow \text{parameters } \lambda \rightarrow \text{model } \hat{\theta}
\]

\[
\hat{\theta} = \arg\min_{\theta \in \Theta} \ell(X, Y, \theta) + \lambda \|\theta\|
\]
Postconditions

\[ \Psi(\hat{\theta}) \]

Examples:

- “the learned model must correctly predict an important item \((\tilde{x}, \tilde{y})\)”

  \[ \hat{\theta}(\tilde{x}) = \tilde{y} \]

- “the learned model must satisfy individual fairness”

  \[ \forall x, x', |p(y = 1 \mid x, \hat{\theta}) - p(y = 1 \mid x', \hat{\theta})| \leq L\|x - x'\| \]
Bug Assumptions

- $\Psi$ satisfied if we were to train through “clean pipeline”
- bugs are changes to the clean pipeline
- $\Psi$ violated on the dirty pipeline
This is not our goal

Just to learn a better model:

$$\min_{\theta \in \Theta} \ell(X, Y, \theta) + \lambda \|\theta\|$$

s.t. $\Psi(\theta) = \text{true}$
This is our goal

To identify bugs and fix them (and learn a better model):

\[
\min_{Y', \hat{\theta}} \| Y - Y' \|
\]

s.t. \( \Psi(\hat{\theta}) = \text{true} \)

\[
\hat{\theta} = \arg\min_{\theta \in \Theta} \ell(X, Y', \theta) + \lambda \| \theta \|
\]
Special case: bugs in training labels

- $\Psi$ satisfied if we were to train on “clean data” $(X, Y')$
- bugs are changes to clean labels

$$(X, Y) = (X, Y' + \Delta)$$

- not just about outliers
- may contain systematic biases
Input / output to our debugger

Input:
1. dirty training set \((X, Y)\)
2. trusted items \((\tilde{X}, \tilde{Y})\)
3. the learner

Output:
1. \(Y'\)
2. confidence
Formulation equivalent to machine teaching

\[
\begin{align*}
\min_{Y'} & \quad \|Y' - Y\| \\
\text{s.t.} & \quad \hat{\theta}(\tilde{X}) = \tilde{Y} \\
\end{align*}
\]

\[
\hat{\theta} = \arg\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} \ell(x_i, y'_i, \theta) + \lambda \|\theta\|^2
\]

Difficult!

- combinatorial
- bilevel optimization (Stackelberg game)

[Dec. 9 Workshop on Teaching Machines, Robots, and Humans]
Combinatorial to continuous relaxation

**step 1.** label to probability simplex

\[ y'_i \rightarrow \delta_i \in \Delta \]

**step 2.** counting to probability mass

\[ \|Y' - Y\| \rightarrow \frac{1}{n} \sum_{i=1}^{n} (1 - \delta_{i,y_i}) \]

**step 3.** soften postcondition

\[ \hat{\theta}(\tilde{X}) = \tilde{Y} \rightarrow \frac{1}{m} \sum_{i=1}^{m} \ell(\tilde{x}_i, \tilde{y}_i, \theta) \]
Continuous now, but still bilevel

\[
\arg\min_{\delta \in \Delta^n, \hat{\theta}} \frac{1}{m} \sum_{i=1}^{m} \ell(\tilde{x}_i, \tilde{y}_i, \hat{\theta}) + \gamma \frac{1}{n} \sum_{i=1}^{n} (1 - \delta_{i,y_i})
\]

s.t. \[
\hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} \delta_{ij} \ell(x_i, j, \theta) + \lambda \|\theta\|^2
\]
Removing the lower level problem

\[ \hat{\theta} = \arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} \delta_{ij} \ell(x_i, j, \theta) + \lambda \|\theta\|^2 \]

**step 1.** the KKT condition

\[ \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{k} \delta_{ij} \nabla_{\theta} \ell(x_i, j, \theta) + 2\lambda \theta = 0 \]

**step 2.** plug implicit function \( \theta(\delta) \) into upper level problem

\[ \arg\min_{\delta} \frac{1}{m} \sum_{i=1}^{m} \ell(\tilde{x}_i, \tilde{y}_i, \theta(\delta)) + \gamma \frac{1}{n} \sum_{i=1}^{n} (1 - \delta_{i,y_i}) \]

**step 3.** compute gradient \( \nabla_{\delta} \) with implicit function theorem
Software available.
Harry Potter Toy Example

- Data
- Our debugger
- Influence function

- Nearest neighbor
- Label noise detection
- Average PR
Another special case: bug in regularization weight

(logistic regression)
Postcondition violated

\[ \Psi(\hat{\theta}) : \text{Individual fairness (Lipschitz condition)} \]

\[ \forall x, x', |p(y = 1 | x, \hat{\theta}) - p(y = 1 | x', \hat{\theta})| \leq L \| x - x' \| \]
Bug assumption

Learner’s regularization weight $\lambda = 0.001$ was inappropriate

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \ell(X, Y, \theta) + \lambda \|\theta\|^2$$
Debugging formulation

$$\min_{\lambda', \hat{\theta}} \quad (\lambda' - \lambda)^2$$

s.t. \quad \Psi(\hat{\theta}) = \text{true}

\[\hat{\theta} = \arg\min_{\theta \in \Theta} \ell(X, Y, \theta) + \lambda'\|\theta\|^2\]
Suggested bug

\[ \lambda = 0.001 \rightarrow \lambda' = 121 \]
Call for ML bug repository

- like software bug repositories in software engineering
- need data provenance
  - which training items (or other things) were wrong
  - what they should be
References

- Xuezhou Zhang, Xiaojin Zhu, and Stephen Wright. Training set debugging using trusted items. AAAI 2018
- Shalini Ghosh, Patrick Lincoln, Ashish Tiwari, and Xiaojin Zhu. Trusted machine learning for probabilistic models. –
  http://www.cs.wisc.edu/~jerryzhu/machineteaching/