Slice Finder: Automated Data Slicing for Model Interpretability

Yeounoh Chung, Tim Kraska
Brown University
{yeounoh_chung,tim_kraska}@brown.edu

Steven Euijong Whang*, Neoklis Polyzotis
Google Research
{swhang,npolyzotis}@google.com

ABSTRACT

As machine learning (ML) systems become democratized, helping users easily debug their models becomes increasingly important. Yet current data tools are still primitive when it comes to helping users trace model performance problems all the way to the data. We focus on the particular problem of slicing data to identify subsets of the training data where the model performs poorly. Unlike general techniques (e.g., clustering) that can find arbitrary slices, our goal is to find interpretable slices (which are easier to take action compared to arbitrary subsets) that are problematic and large. We propose Slice Finder, which is an interactive framework for identifying such slices using statistical techniques. The slices can be used for applications like diagnosing model fairness and fraud detection where describing slices that are interpretable to humans is necessary.

1 INTRODUCTION

Machine learning (ML) systems [3] are becoming more prevalent thanks at least in-part due to improvements in state-of-the-art performance on common tasks. However, the data tools for interpreting and debugging models have not caught up yet [5], and ML training tends to be a black box process requiring trial and error until the model performs satisfactorily. The problem becomes increasingly difficult as the size of data increases. The challenges for model interpretability are broad. One example is measuring feature attribution to model performance (e.g., how much impact does feature foo have on my deep model?). Another example is measuring model fairness (e.g., is the model discriminating a certain demographic?).

An important challenge in model interpretation is which slices of the data are causing the model to have poor metric values (e.g., a high loss) making them problematic. The challenge is that, although the model may appear to be performing well overall, it may not on smaller data slices [8]. To capture the model's behavior on slices, ML practitioners commonly use a few manually-defined slices to avoid serving models that sacrifice the model performance on these slices for better overall performance. However, this approach requires domain expertise to find the right slices and may not scale if there are many features. In addition, the number of possible slices is exponential to the number of examples, making an exhaustive approach of considering all slices infeasible as well.

Slice Finder addresses the data slicing problem by applying data management techniques to efficiently discover the top-K largest slices that are *interpretable* and *problematic*. The interpretability is necessary to provide insight to the user and enable further action (e.g., cleaning the data). For example, a slice of customers who are female and live in India (expressed as gender = Female $\wedge$ country = IN) is more interpretable than a cluster of 10 people $\{x_1, x_2, \cdots, x_{10}\}$. A slice also needs to be problematic in the sense that its effect size (defined in Section 2 and captures how differently the model performs compared to other data) is larger than a threshold that can be adjusted. Slice Finder can be used as a preprocessing step that guides the user to narrow down on slices with the above properties on which more sophisticated analyses (e.g., measuring model fairness [6]) can be performed.

2 PROBLEM DEFINITION

We assume a dataset D containing examples and a model M. Each example contains features (e.g., country) where each feature has a list of values (e.g., {US, DE}). We denote the entire set of possible features as F. A model is a function that receives an example and outputs a prediction. We also assume a metric ψ that returns a score for each example by comparing M's prediction p with the example's label y. For example, if ψ measures log loss, it is $y \log p + (1-y) \log (1-p)$.

A *slice* S is a subset of examples in D with common features and can be described as a conjunction of the common feature-value pairs $\bigwedge_i F_i = v_i$ where the F_i's are distinct (e.g., country = DE $\wedge$ gender = Male). For numeric features, we discretize their values and generate ranges so that they are effectively categorical features (e.g., age = 20-30).

A slice is *problematic* if the metric values between the slice and its counterpart slice have an effect size larger than a given threshold T. The definition of a counterpart slice depends on the problem being solved. If we are evaluating a single model, then the counterpart can be defined as the slice containing all the other examples not in S. If two models are being compared, the counterpart is the same slice S, but using the other model. If there is one model, but two model serving datasets are being compared, the counterpart is the slice in the other dataset. The effect size [1] is a standardized mean difference between two populations (e.g., gender = Male vs. gender = Female):

*Corresponding author

SysML, February 2018, Stanford, California USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

$$2 \times \frac{\mu_S - \mu_{S^c}}{\sigma_S + \sigma_{S^c}} \quad (1)$$

where μ_S is the mean value of ψ for S, σ_S is the standard deviation, and S^c is the counterpart slice of S. The threshold T can be set using known conventions (e.g., Cohen's convention [4] considers 0.2 to be small, 0.5 medium, 0.8 large, and 1.3 very large). While effect size can be replaced with any other measure, it is good at measuring the size of the difference regardless of slice sizes.

When measuring interpretability, we assume that the fewer the features crossed for describing a slice, the more interpretable. For example, country = DE is more interpretable than country = DE $\wedge$ age = 20-40 $\wedge$ zip = 12345. More sophisticated measures (e.g., taking into account the semantics of features) can be used as well.

Definition 2.1. The data slicing problem is defined as finding the top-K largest slices such that:

- Each slice has an effect size at least T and
- No slice can be replaced with another with the same size, but with fewer features.

Note that the top-K slices do not have to be distinct, e.g., country = DE and education = Bachelors overlap in Germans with a Bachelors degree.

3 SLICE FINDER

We now describe the Slice Finder system. The input is the training data, a model, and an effect size threshold T. As a preprocessing step, Slice Finder takes the training data and discretizes numeric features. For categorical features that contain too many values (e.g., IDs are unique for each example), Slice Finder uses a heuristic where it considers up to N most frequent values and places the rest into an "other values" bucket. The possible slices of these features form a lattice where a slice S is a parent of every S with exactly one more feature-value pair.

Slice Finder finds the top-K largest problematic slices by iterating the slice lattice in a breadth-first manner using a priority queue. The priority queue contains the current slices being considered sorted by descending size and then by ascending number of features. For each slice $\bigwedge_{i \in I} F_i = v_i$ that is popped, Slice Finder checks if it has an effect size at least T. If so, the slice is added to the top-K list. Otherwise, the slice is expanded where the slices $\{\bigwedge_{i \in I} F_i = v_i \wedge G = v | G \in F - \{F_1, \ldots, F_{|I|}\}, v \in G's \text{ values}\}$ are added to the queue. Slice Finder optimizes this traversal by avoiding slices that are subsets of previously generated slices. This process repeats until either the top-K slices have been found or there are no more slices to explore.

Example. Suppose there are three features A, B, and C with the possible values $\{a_1\}$ and $\{b_1, b_2\}$, and $\{c_1\}$, respectively. Also say $K = 2$, and the effect size threshold is T. Initially, the priority queue Q contains the entire slice. This slice is popped and expanded to the slices $A = a_1$, $B = b_1$, $B = b_2$, and $C = c_1$, which are inserted back into the queue. Among them, suppose $A = a_1$ is the largest slice with an effect size at least T. Then this slice is popped from Q and added to the top-K result. Suppose that no other slice has an effect size at least T, but $B = b_1$ is the largest. This slice is then expanded to $B = b_1 \wedge C = c_1$ (notice that $B = b_1 \wedge A = a_1$ is unnecessary because it is a subset of $A = a_1$). If this slice has an effect size at least T, then the final result is $[A = a_1, B = b_1 \wedge C = c_1]$.

We can show that the Slice Finder slices satisfy Definition 2.1 using proof-by-contradiction.

Slice Finder also provides a configurable slider for adjusting T, which can be done through a UI. If T decreases, then we just need to reiterate the slices explored until now to find the top-K slices. If T increases, then the current slices may not be sufficient, so we continue searching the slice lattice.

Slice Finder can be used along with visualization tools for the slices. State-of-art tools include Facets [2], which can be used to discover bias in the data, and MLCube [7], which provides manual exploration of slices and can both evaluate a single model or compare two models. While the above tools are manual, Slice Finder complements them by automatically finding slices.

4 USE CASES

We present some motivating applications for Slice Finder .

Model Fairness. Due to biases in training data, models can perform worse on certain demographics. For sensitive attributes (e.g., gender), such difference can be viewed as discrimination. The interpretable slices produced by Slice Finder combined with model fairness metrics [6] can be used to quickly identify parts of the training data that need to be augmented with more data collection.

Fraud Detection. Finding fraudulent users involves identifying demographics where a model is not performing as well as it previously did. For example, some fraudsters may have gamed the system with unauthorized transactions. Here Slice Finder can be used to identify the slices for two datasets where the same model performs most differently.

Anomaly Detection. Although Slice Finder assumes a model, the data slicing problem can be more general where it uses any scoring function instead of a model. For example, the scoring function can reflect the number of anomalies and take into account temporal aspects such as how recently they occurred. Finding anomalies in data and identifying slices that contain relatively more anomalies is important for ensuring data quality in ML pipelines.

REFERENCES

[1] 2017. Effect Size. (2017). https://en.wikipedia.org/wiki/Effect_size
[2] 2017. Facets Overview. (2017). https://research.googleblog.com/2017/07/facets-open-source-visualization-tool.html
[3] Denis Baylor, Eric Breck, Heng-Tze Cheng, Noah Fiedel, Chuan Yu Foo, Zakaria Haque, Salem Haykal, Mustafa Ispir, Vihan Jain, Levent Koc, et al. 2017. TFX: A TensorFlow-Based Production-Scale Machine Learning Platform. In *Proc. of ACM SIGKDD*. 1387–1395.
[4] Jacob Cohen. 1988. Statistical power analysis for the behavioral sciences . Hilsdale. *NJ: Lawrence Earlbaum Associates* 2 (1988).

[5] F. Doshi-Velez and B. Kim. 2017. Towards A Rigorous Science of Interpretable Machine Learning. *ArXiv e-prints* (Feb. 2017). arXiv:stat.ML/1702.08608

[6] Moritz Hardt, Eric Price, and Nati Srebro. 2016. Equality of Opportunity in Supervised Learning. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. 3315–3323. http://papers.nips.cc/paper/6374-equality-of-opportunity-in-supervised-learning

[7] Minsuk Kahng, Dezhi Fang, and Duen Horng Polo Chau. 2016. Visual exploration of machine learning results using data cube analysis. In *Proceedings of the Workshop on Human-In-the-Loop Data Analytics*. ACM, 1.

[8] H Brendan McMahan, Gary Holt, David Sculley, Michael Young, Dietmar Ebner, Julian Grady, Lan Nie, Todd Phillips, Eugene Davydov, Daniel Golovin, et al. 2013. Ad click prediction: a view from the trenches. In *Proc. of ACM SIGKDD*. 1222–1230.