Combinatorial Testing Metrics for Machine Learning

Erin Lanus*, Laura J. Freeman*, D. Richard Kuhn†, Raghu N. Kacker†
*Hume Center for National Security and Technology, Virginia Tech, Arlington, VA, USA
{lanus, laura.freeman}@vt.edu
†National Institute of Standards and Technology, Gaithersburg, MD, USA
{kuhn, raghu.kacker}@nist.gov

***Abstract*—This short paper defines a combinatorial coverage metric for comparing machine learning (ML) data sets and proposes the differences between data sets as a function of combinatorial coverage. The paper illustrates its utility for evaluating and predicting performance of ML models. Identifying and measuring differences between data sets can be of significant value for ML problems, where the accuracy of the model is heavily dependent on the degree to which training data are sufficiently representative of data that will be encountered in application. The utility of the method is illustrated for transfer learning, the problem of predicting performance of a model trained on one data set when applied to another.**

***Index Terms*—combinatorial testing, machine learning, operating envelopes, transfer learning, test-set selection**

I. Introduction

In software and hardware, component systems are often well designed and tested, but failures occur when components are integrated due to unexpected interactions between components. A survey [1] of empirical data found that nearly all failures in software were caused by a limited number of interacting components and concluded that testing interactions of between four and six components was sufficient to detect all failures in the software systems considered. The pair of facts – that interactions between components called *factors* in a complex system can drive unexpected behavior if tests do not adequately account for interactions, yet impactful interactions typically have limited size – has led to the adoption of combinatorial testing (CT) for pseudo-exhaustive testing of software and hardware systems [2].

Testing systems with embedded machine learning using conventional software approaches poses challenges due to characteristics such as the large input space, effort required for white box testing, and emergent behaviors apparent only at integration or system levels [3], [4]. CT is a black box approach to testing an integrated system with pseudo-exhaustive strategy for large input spaces. Thus far, CT has been applied to test case generation for autonomous vehicle systems with embedded ML components [5], testing the internal state space of a neural network [6], feature selection [7], and explainable ML [8].

In ML the data is fundamental to algorithm performance. In this paper we leverage CT for testing ML systems through comparison of datasets. Consider the following questions. How can differences between members of two classes lead to classification decisions? How can differences between datasets be used to predict whether a model trained on one dataset will perform as expected on another? Comparing datasets via combinations is possible at three levels of granularity: 1) the count of combinations that are present or absent; 2) which specific combinations are present or absent; 3) the distribution of combinations.

In this work, we define a new combinatorial coverage metric for comparing ML datasets in § II focusing on the first level of granularity (presence/absence of combinations). In § III, we highlight two distinct applications of the metric. The fault localization and explainable classification sections show how the metric is useful based on interpretable features in the data. The second application use the metrics to describe the operating envelope of a model. The concept of a model operating envelope has applications in transfer learning, selection of training and test datasets, and directing data collection and labeling efforts. We discuss problems for future work in § IV.

Research of EL and LJF funded in part by MITRE University Innovation Exchange program.

II. Metrics

We treat machine learning *features* as factors, so each factor is assigned a particular value for a given data point. Continuous-valued factors must be discretized prior to applying CT so that each factor has a corresponding set of *values*. A t-way interaction is an assignment of specific values to t of the factors, or a t-tuple of (factor, value) pairs. If there are k identified features, each data point then contains $\binom{k}{t}$ interactions.

Combinatorial coverage, also called total t-way coverage, is a metric from the CT literature [9] to describe the proportion of possible t-way interactions appearing in a set (Figure 1). Interactions that appear in the set are *covered* by the set. Define a universe with k factors and their respective levels so that $\mathcal{U}$ is the set of all possible datapoints, and let $\mathcal{U}_t$ be the set of possible t-way interactions. If some interaction is not possible, it is a *constraint* and can be removed from $\mathcal{U}_t$. Given a dataset $\mathcal{D} \subseteq \mathcal{U}$, define $\mathcal{D}_t$ as the set of t-way combinations appearing in $\mathcal{D}$. (We acknowledge a slight abuse of notation as $\mathcal{D}$ may be a multiset. This does not impact the metrics.)

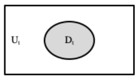

Fig. 1. $CCM_t(\mathcal{D}_t)$ describes the proportion of possible t-way interactions that appear in dataset $\mathcal{D}$.

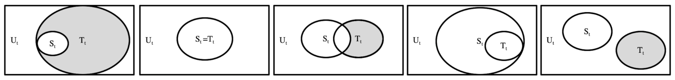

Fig. 2. Relationships between datasets $\mathcal{S}$ and $\mathcal{T}$ and corresponding $SDCCM_t$ values. 1) $(\mathcal{S}_t \subset \mathcal{T}_t), 0 < SDCCM_t(\mathcal{T}_t \setminus \mathcal{S}_t) < 1$; 2) $(\mathcal{S}_t = \mathcal{T}_t), SDCCM_t(\mathcal{T}_t \setminus \mathcal{S}_t) = 0$; 3) $(\mathcal{S}_t \not\subset \mathcal{T}_t) \wedge (\mathcal{T}_t \not\subset \mathcal{S}_t) \wedge (\mathcal{S}_t \bigcap \mathcal{T}_t \neq \emptyset), 0 < SDCCM_t(\mathcal{T}_t \setminus \mathcal{S}_t) < 1$; 4) $(\mathcal{T}_t \subset \mathcal{S}_t), SDCCM_t(\mathcal{T}_t \setminus \mathcal{S}_t) = 0$; 5) $(\mathcal{S}_t \bigcap \mathcal{T}_t) = \emptyset), SDCCM_t(\mathcal{T}_t \setminus \mathcal{S}_t) = 1$.

The combinatorial coverage metric of $\mathcal{D}$ is

$$CCM_t(\mathcal{D}_t) = \frac{|\mathcal{D}_t|}{|\mathcal{U}_t|}$$

where $|\mathcal{D}_t|$ denotes set cardinality.

Let $\mathcal{S}$ and $\mathcal{T}$ be datasets and define $\mathcal{S}_t, \mathcal{T}_t$ as the set of t-way interactions appearing in $\mathcal{S}, \mathcal{T}$, respectively. The set difference $\mathcal{T}_t \setminus \mathcal{S}_t$ is the set of interactions appearing in $\mathcal{T}_t$ but not in $\mathcal{S}_t$. Then the set difference combinatorial coverage

$$SDCCM_t(\mathcal{T}_t \setminus \mathcal{S}_t) = \frac{|\mathcal{T}_t \setminus \mathcal{S}_t|}{|\mathcal{T}_t|}$$

is the proportion of t-way interactions appearing in $\mathcal{T}$ but not $\mathcal{S}$. Constraints, or impossible interactions, need not be explicitly defined as only interactions present in $\mathcal{T}$ are considered. $SDCCM_t$ is a score between 0 and 1 inclusive reflecting the five cases of relationship between $\mathcal{T}_t$ and $\mathcal{S}_t$ depicted in Figure 2. As a difference metric, higher values indicate that the set difference is larger.

At the coarsest level of granularity, coverage is represented as a single score or venn diagram as shown for CCM_t and $SDCCM_t$. To provide more information, the interactions not appearing in $\mathcal{D}_t$ for CCM_t and interactions in the set difference $\mathcal{T} \setminus \mathcal{S}$ for $SDCCM_t$ are listed or plotted as status per interaction. A heatmap of relative frequency of interactions for CCM_t and difference in relative frequency for $SDCCM_t$ provides the finest granularity level of analysis.

III. Applications

A. Fault localization

Set differencing of t-way interactions has been applied to the problem of fault localization. A variety of set theoretic operations can be used in reducing the set of possible failure-triggering combinations in deterministic software [10]. Running a test set typically results in a large number of passing tests and a small number of failing tests, but only a small subset of combinations in the failing tests will induce a failure. For P_t = combinations in passing tests and F_t = combinations in failing tests and C_t = fault-triggering combinations, the first step in identifying failure-triggering combinations is a basic elimination rule: compute $F_t \setminus \mathrm{P}_t$, combinations in failing tests that are not in any passing tests, which for deterministic systems must contain the fault-triggering combinations C_t. Basic set operations can also be used to further reduce the possible combinations involved in a failure. For example, an interaction continuity rule says that if a particular t-way combination in F_t is included in all higher strength combinations that contain the same t parameters, then the t-way combination is sufficient to detect the error.

B. Explainable Classification

From a certain perspective, the problem of classification in ML is essentially the same as the fault localization problem in CT. We seek to identify a small subset of factors that distinguish the class from instances not in the class. This process could be viewed as generalizing the fault localization problem, where the failing tests are the class and passing tests are non-class members - what combinations of parameter values are unique to the failing tests?

This simple observation leads to a method of producing explanations or justifications of ML classifications [11], by computing $C_t \setminus \mathrm{N}_t$, the set of t-way combinations that appear in members of the class C which are not in the non-class members of N, or are more strongly associated with C than N. For example, applying this method in a database of animal characteristics produces seven predicates that are unique to reptiles (within this database): *not aquatic AND not toothed AND four legs egg-laying AND not aquatic AND four legs*, etc. These combinations have an obvious mapping with simple rules: "if non aquatic AND not toothed AND … ". No single-factor or 2-way combinations are uniquely associated with the reptile class, but including 3-way combinations makes it possible to identify class members.

Previous model induction methods have been developed to reverse engineer an explanation or model from ML output [12], [13], using statistical methods to identify characteristics most closely associated with a class. The combinatorial XAI method extends this approach by producing combinations of characteristics for explanation. This distinction is important because closely associated single factors are not necessarily contained in identifying combinations. Rule-based expert systems are often considered easy to explain but generally are not as proficient as more opaque methods such as neural networks [14]. The combinatorial approach to XAI provides a natural mapping to clearly understandable diagnostic rules.

C. Model Operating Envelope

Computer vision includes tasks such as detecting or classifying an object in an image. The complexity of the domain – all of the variables affecting the production of an image – leads to high likelihood of interaction effects. Consider the problem of detecting a white truck in an image. A white truck against a light background at noon from an overhead view likely presents a more difficult detection scenario than a white truck against the same light background in late afternoon where shadows are present, or from profile such that the horizon line breaks up the background. The operating envelope of an ML model describes the contexts in which it is expected to perform correctly; deploying to contexts outside of the envelope can lead to unexpected outcomes. An ML model learns about examples on which it trains, so to perform as expected in each of these contexts, it is anticipated that "enough" representative examples must be included in the training dataset. The challenge is how to define contexts and measure representativeness of the training examples.

One dimension of the operating envelope of a computer vision algorithm describes the contexts in which the model trained as coverage of interactions among features present in the dataset. These features can be derived directly from the image data, but there are two benefits of using metadata such as "Time of Day" or "Location" collected along with the image acting as a surrogate for contexts present in the image. metadata is likely easier for human operators; "Time of Day" as a surrogate for lighting effects in the image is more quickly understood than presenting values for luminance and contrast. metadata may be available when image data is not, such as the case when an event is occurring in the near future in a new deployment environment for which no images have been collected. Expected parameters such as "Time of Day" and "Location" can be extracted from the event profile.

When class labels are available, we describe a special way of calculating interactions. *Label centrism* forces all interactions to include the label; that is, a label-centric interaction includes the label and $t-1$ of the other features. Label centrism describes the contexts in which classifiable objects appear.

In the current practice, claiming representativeness of a training dataset often relies on randomized selection or ensures that every object type appears in the training set, but may fail to be representative of larger contexts of the deployment environment. CCM_t applied to a training dataset provides a measurement of the contexts on which the algorithm trained given the tunable parameter t. In the case of transfer learning, a model trained in one environment is deployed to a different environment, possibly without retraining or fine tuning. Where CCM_t is a measure of coverage by a dataset with respect to some defined universe, the new metric, $SDCCM_t$, describes a directed difference between two datasets and is useful for measuring the distance between a source dataset where the model is trained and a target dataset $\mathcal{T}$ where the model will be deployed. When multiple source models are available in a *model zoo*, the source dataset $\mathcal{S}$ with the smallest $SDCCM_t(\mathcal{T} \setminus \mathcal{S})$ provides the best coverage of contexts in the target by the source (Figure 3). Additionally, as interactions in a set difference describe contexts unseen in the trained model, the list of interactions in the set difference provides a mechanism for directing data collection or labeling efforts to include datapoints containing these interactions.

A use case for the set difference application to operating envelopes for transfer learning is demonstrated on the "Planes in Satellite Imagery" Kaggle dataset [15]. The dataset is intended for binary classification and is comprised of images that either have a plane or do not have a plane along with metadata indicating the location as Northern California or

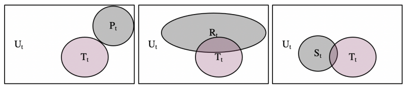

Fig. 3. Metadata interactions expected in the target set can be used to select a source set from the model zoo with the closest match on metadata interactions

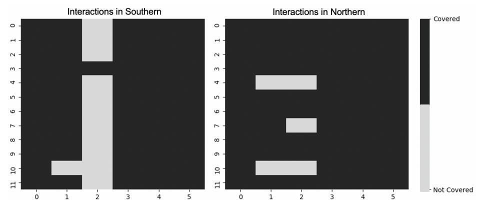

Fig. 4. Covered and uncovered 2-way label-centric interactions in the Southern and Northern datasets. The Y-axis indexes into combination while the X-axis indexes into interaction within a combination.

Southern California. When a model is trained on the Southern subset of data $\mathcal{S}$, a performance drop occurs when transferring to the Northern subset of data $\mathcal{T}$, indicating a transfer learning problem; the drop is not noted when the direction of transfer is reversed. Twelve features are derived from the image data – the mean and variance each for the red, green, blue, hue, saturation, and luminance – and each is discretized by forming three bins encompassing equal-sized ranges. Interactions are label-centric and $t = 2$. The Southern set contains 21,151 images and the Northern set contains 10,849 images. The $CCM_2(\mathcal{S}) = \frac{60}{72} = 0.83$ and $CCM_2(\mathcal{T}) = \frac{67}{72} = 0.93$, meaning that the Northern set covers more of the universe than the Southern set despite having half as many images. Figure 4 plots the coverage in the sets side by side.

The utility of CCM for comparing a source and target pair is limited. Suppose $\mathcal{S}'$ contained all interactions in the left half of the plot and none in the right half, while $\mathcal{T}' = \mathcal{U} \setminus \mathcal{S}'$, the complement. Both have CCM_2 values of 0.5. Suppose $\mathcal{S}'' = \mathcal{T}''$ and $CCM_2(\mathcal{S}'') = 0.25$. The relationship between the respective sets is not apparent through use of CCM, which is the limitation for which $SDCCM$ was designed. $SDCCM_2(\mathcal{S} \setminus \mathcal{T}) = \frac{1}{60} = 0.02$. $SDCCM_2(\mathcal{T} \setminus \mathcal{S}) = \frac{8}{67} = 0.12$. For this dataset, $SDCCM$ is correlated with a drop in performance in transfer learning when no retraining is allowed. Figure 5 depicts the $SDCCM_2$ plots side by side.

Combinatorial coverage is well studied in testing for deterministic failures in software systems where the appearance of an interaction among components in one test is sufficient

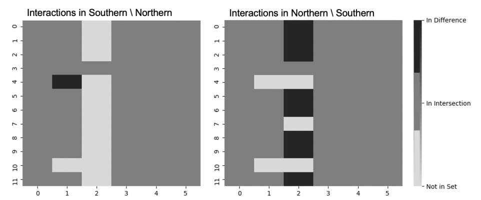

Fig. 5. Set differences of 2-way label-centric interactions. Interactions appear as cells colored depending set membership.

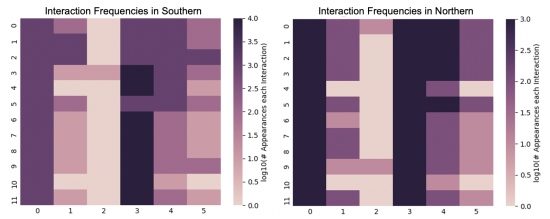

Fig. 6. The log scale count of each interaction in each set.

to cause a failure; if the components will interact to cause a failure, it is detected by a test suite that contains that interaction at least once. Statistical learning does not have this nice property. We suspect that combinatorical coverage and set difference combinatorial coverage metrics are useful tools for constructing operating envelopes as they provide a way to detect contexts in the target environment that are not likely to be within the model's operating envelope. However, as models are trained by updating weights each time these contexts are seen, we suspect that distribution of coverage would do more to describe the operating envelope over metadata. That is, frequently appearing interactions indicate contexts on which the model was well trained; they could also indicate instances of overfitting. Infrequently appearing interactions indicate contexts on which the model received less training; they could present contexts in which the model has difficulty making classifications. Our work measures and plots this distribution.

D. Test Set Design

Datasets are partitioned into training $\mathcal{S}$, validation, and testing $\mathcal{T}$ sets. When datasets are large and random selection is applied, the hope is that the test set is representative of the training set as they are drawn from the same population. Computing $SDCCM_t(\mathcal{S}\setminus\mathcal{T})$ and $SDCCM_t(\mathcal{T}\setminus\mathcal{S})$ provides assurance against a bad random draw. A simple randomized algorithm makes several random partitions and keeps the one with the lowest $SDCCM$ values. This is equivalent to testing within the operating envelope of the model.

Another testing strategy is identify where the model fails to generalize to new contexts it has not trained by selecting test-sets outside of the envelope. In this case, selecting $\mathcal{T}$ so that $SDCCM_t(\mathcal{T}\setminus\mathcal{S})$ is as close to 1 as possible creates a test set that contains as many untrained contexts as possible. The importance of the reverse direction is not as clear. When $SDCCM_t(\mathcal{T}\setminus\mathcal{S}) = 1$, the sets T_t and S_t are disjoint and $SDCCM_t(\mathcal{S}\setminus\mathcal{T}) = 1$ necessarily, but when it is only close to 1, the score also depends on the size of S_t.

IV. Conclusions and Future Work

This work discussed metrics that provide tools for explaining classification outcomes and defining the domain (operating envelope) over which ML algorithms can be expected to operate successfully. Future work is needed to explore the usefulness of these metrics across multiple ML domains, test the hypothesis that models trained on source sets with smaller $SDCCM_t$ distances to the target will perform better in the target environment, explore the impact of label centrism, and how to choose a "good" interaction size t.

Additionally, the sensitivity of these metrics to feature/metadata selection is critical. In the classification example, the features were directly explainable. In the computer vision example the research had to first hypothesize reasonable features. The process of hypothesizing features, conducting initial screening experiments to select the meaningful features, and confirming results should be codified to ensure that this work is not subject to confirmation biases of the research team or over interpretation of correlations as explanatory variables.

Finally, additional work is needed to exploit the deeper levels of explainability, that is - which specific interactions are present or absent and the distribution of those interactions. The specific interactions present or absent should be explored for potential explanation of how and why algorithms perform well or poorly, potential biases introduced into the algorithms, and predictive capabilities to new operating envelopes. Relative frequency metrics for the set difference should be developed and their application to transferability evaluated.

References

[1] D. R. Kuhn, D. R. Wallace, and A. M. Gallo, "Software fault interactions and implications for software testing," *IEEE Transactions on Software Engineering*, vol. 30, no. 6, pp. 418–421, 2004.

[2] C. Nie and H. Leung, "A survey of combinatorial testing," *ACM Computing Surveys (CSUR)*, vol. 43, no. 2, pp. 1–29, 2011.

[3] D. Marijan, A. Gotlieb, and M. Kumar Ahuja, "Challenges of testing machine learning based systems," in *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*, 2019, pp. 101–102.

[4] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine learning testing: Survey, landscapes and horizons," *IEEE Transactions on Software Engineering*, pp. 1–1, 2020.

[5] C. E. Tuncali, G. Fainekos, H. Ito, and J. Kapinski, "Simulation-based adversarial test generation for autonomous vehicles with machine learning components," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1555–1562.

[6] L. Ma, F. Juefei-Xu, M. Xue, B. Li, L. Li, Y. Liu, and J. Zhao, "Deepct: Tomographic combinatorial testing for deep learning systems," in *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 2019, pp. 614–618.

[7] S. Vilkomir, J. Wang, N. L. Thai, and J. Ding, "Combinatorial methods of feature selection for cell image classification," in *2017 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, 2017, pp. 55–60.

[8] R. Kuhn and R. Kacker, "An application of combinatorial methods for explainability in artificial intelligence and machine learning (draft)," National Institute of Standards and Technology, Tech. Rep., 2019.

[9] D. R. Kuhn, I. D. Mendoza, R. N. Kacker, and Y. Lei, "Combinatorial coverage measurement concepts and applications," in *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops*, 2013, pp. 352–361.

[10] D. R. Kuhn, R. N. Kacker, and Y. Lei, "Practical combinatorial testing," *NIST special Publication*, vol. 800, no. 142, p. 142, 2010.

[11] D. R. Kuhn, R. N. Kacker, Y. Lei, and D. E. Simos, "Combinatorial methods for explainable ai."

[12] M. T. Ribeiro, S. Singh, and C. Guestrin, "" why should i trust you?" explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 2016, pp. 1135–1144.

[13] F. Shakerin and G. Gupta, "Induction of non-monotonic logic programs to explain boosted tree models using lime," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3052–3059.

[14] D. Gunning, "Explainable artificial intelligence (xai)," *Defense Advanced Research Projects Agency (DARPA), nd Web*, vol. 2, no. 2, 2017.

[15] Rhammell, "Planes in satellite imagery," Jan 2018. [Online]. Available: https://www.kaggle.com/rhammell/planesnet