Aloha: A Machine Learning Framework for Engineers

Ryan M Deak
ZEFR, Inc.
ryan.deak@zefr.com

Jonathan H Morra
ZEFR, Inc.
jon.morra@zefr.com

ABSTRACT

The process of deploying machine learnt models to production systems can be difficult, time consuming, and prone to many engineering and data-related risks. Here, we present Aloha[1], a model representation framework that mitigates many of these risks. In Aloha, feature expansion and reporting are subsumed within the model boundary, allowing Aloha to read *user-defined*, rather than *framework-defined* data types, which eases integration in preexisting engineering pipelines. Aloha's reporting functionality has a variety of use cases including data anomaly detection and hierarchical model output logging. The latter benefit enables simple A/B testing and is also useful in classification scenarios when offline analysis on raw model scores is desirable. Feature expansion and reporting are performed non-strictly to avoid unnecessary computation.

1 INTRODUCTION

The difficulty of deploying and maintaining machine learning (ML) pipelines to create and quickly productionalize models is often a stumbling block to those wishing to make an investment in ML. Getting even simple, more well understood models such as linear models [32], decision trees [4], and boosting models [9] into production can be a high cost endeavor. Many of the common pitfalls are outlined by Sculley *et al.* [25].

At the heart of many of these problems is a fundamental difference in the domains on which data scientists and engineers operate. Data scientists utilize many ML packages operating on library-specific representations of features, typically encoded as tensors. These representations are often transformed and flattened views of structured data. Data engineers are responsible for supplying data to various teams within an organization and they may prefer to store, retrieve, and operate on structured data as is evidenced with the proliferation of NoSQL databases [30] like MongoDB [7], Cassandra [20] and Redis [5].

To complicate matters further, data requirements vary between teams serviced by data scientists. This can be compounded by organizational fragmentation. To facilitate the use of ML models in such environments, a principled methodology for interacting with models is imperative. A common strategy is to wrap models in microservices. In systems that use batch processing, this can be difficult to scale. To overcome these cases, model exchange formats like PMML [11], PFA [24] and ONNX [3] can be used. The benefit of these frameworks, as *Bird et al.* [3] point out, is that an intermediate representation (IR) for models decreases the complexity of the model translation problem from $O(M^2)$ to $O(M)$, where M is the number of ML packages.

Aloha is a framework that acts less as an intermediary between multiple ML packages, like ONNX, and more like a bridge between data representations and ML packages. So, analogously, given N input data representations, K output data representations and, M ML packages, Aloha decreases complexity of applying ML packages to data representations from an $O(NMK)$ problem to an $O(N+M+K)$ problem through its use of an IR. This is not a novel idea in ML. Many ML packages implement multiple algorithms with diverse structure through a canonical data representation. WEKA provided this functionality over two decades ago [13]. Instead of introducing another data representation, Aloha's representation is a pair of *descriptions* for extracting and transforming data from a desired model domain and exporting predictions to a desired codomain. Users create descriptions of features and output transformations that are applied non-strictly. For wrapped black-box ML packages, Aloha applies these at the model boundary of the wrapped model. For models implemented natively in Aloha, feature expansion and output transformation is more tightly integrated to promote non-strictness, which is useful in decision trees and anytime algorithms.

2 RELATED WORK

Many ML packages provide features found in Aloha. TensorFlow [1] supports many algorithms, allows feature expansion inside model definitions, and has control flow to skip subgraph computations. During inference, a typical pattern is to transform input maps of *name-feature list* pairs to internal tensor representation via dataset-specific callbacks [28].

Uber's internal ML platform, Michelangelo [12], wraps many ML packages including MLLib [23], XGBoost [6], and TensorFlow and includes a domain specific language (DSL) for feature selection and transformation, implemented as sub-set of Scala [27]. Michelangelo is currently closed-source and details of its data representation or DSL have not, to the best of the authors' knowledge, been disclosed publicly.

Amazon's SageMaker [17], like Michelangelo, delegates learning and inference to its supported ML packages. SageMaker supports multiple input formats; however, each wrapped

[1] Code is available at https://github.com/eHarmony/aloha under MIT License.

ML package supports a different subset of such formats [16]. Supporting multiple input formats makes it easy to wrap additional ML packages, but may be onerous to callers who must coordinate between input formats during inference calls [18]. Training data for SageMaker must exist on Amazon's S3 storage and be in an ML package-specific format. If data already stored on S3 has an incompatible format, an extract, transform, load (ETL) process is needed to write data back to S3 [19].

3 ARCHITECTURE

Aloha provides a specification format for dataset generation and model inference and interpreters to build ML pipelines. At train time, Aloha's dataset generation functionality transforms data with user-defined schemas to the input format of a wide variety of ML frameworks. Aloha's feature expansion DSL is based on Scala and supports the use of arbitrary user-defined functions. Once a dataset is produced and a model is trained, the trained model along with the feature specifications are embedded in a model specification that is interpreted to create an Aloha model used for inference.

Aloha supports a variety of model types including native support for linear models, decision trees, and decision trees with other *submodels* at the nodes. Decision trees of submodels allow granular reporting; data can be tracked as it passes through each submodel. Aloha also wraps other ML packages including Vowpal Wabbit [21] (VW) and H2O.ai [26]. **Listing 1** shows an Aloha model specification that wraps a VW model.

```
{ "modelType":"VwJNI",
  "modelId": { "id": 1, "name": "title model" },
  "vw": { "params":   "-t --quiet",
          "modelUrl": "s3://bucket/abc123.vw", "via": "vfs2" },
  "features": { "title":  "skipGrams(${text.title}, n=2, k=1)",
                "n_tags": "${tags}.length" },
  "namespaces": { "A": ["title"], "B": ["n_tags"] }
}
```

Listing 1: Sample Aloha VW model specification

Wrapped models can be imported from an external source as in **Listing 1** or defined inline. Metaparameters are provided for the configuration of the wrapped model. Perhaps more importantly, variables are encoded using quasiquotation [2] and types are omitted from feature definitions, yet strong, static typing is employed.

When a model specification is interpreted, type information about the feature expansion functions is merged with type information about the input. Type inference aids proper typing without the need for explicit type annotations. Data extraction and transformation are decoupled. Given a feature specification defined in terms of one variable U, a data extractor function $u : \mathcal{X} \to \mathcal{U}$ is created along with an aggregator $a : \mathcal{U} \to B$. The feature expansion function is then $f_1 : \mathcal{X} \to B$ defined as $f_1(x) = a(u(x))$. Higher arity functions can be defined similarly, *e.g.*, $f_2(x) = a(u(x), v(x))$. As data is extracted from the input, missing and anomalous data are aggregated by the model and possibly returned along with the model prediction.

When a model interpreter is created, an *auditor* is supplied to establish the prediction format and diagnostic information to be returned by the model. Examples include returning a single scalar prediction with no diagnostics, to returning a typed tree of model and submodel predictions with model identifier information and diagnostics about missing features and anomalies encountered during inference. The former might be applicable for batch jobs whereas the latter is appropriate when more detail is desirable, *e.g.*, in asynchronous *fire-and-forget* architectures [33].

4 USE CASES

Aloha's non-strict feature expansion is an asset when CPU costs are constrained during inference as in Xu *et al.* [31]. Prediction with constrained CPU costs has also been considered for linear models [15], boosting [10] and neural networks [14].

Decision trees of submodels have also been used to silo models trained on mutually exclusive datasets, where classification based on certain features is disallowed by law in particular locales. Here, strict feature expansion leads to wasted computation in locales where those features cannot be used.

Aloha supports $O(1)$ pseudo-random splits in decision trees and sampling from categorical distributions via salted hashing, which has been used for A/B testing and exploration in discrete action spaces [22]. In these situations, Aloha's auditing capability is important for recording which submodel generated the prediction. Since auditing is performed inside the model, the only configuration change needed is the type of auditor supplied to the model interpreter.

5 DISCUSSION

Aloha supports a number of protocol-based serialization formats including Protocol Buffers [29] and Avro [8] and consumes data with *user-defined* schemas. It wraps black-box ML packages and includes native models to promote non-strictness. Finally, it has flexible support for returning various amounts of diagnostics along with predictions.

By subsuming feature expansion and reporting inside the model boundary, Aloha avoids a canonical data representation at the API level but provides much of the benefit. Users are able to provide and consume their own desired data types and data is extracted and transformed only as needed, which facilitates the use of anytime algorithms in which feature expansion should be performed parsimoniously. Since input types, ML packages and auditors are decoupled, it is easy to add implementations of any of the components without affecting the others.

REFERENCES

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Gregory S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian J. Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Józefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Gordon Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul A. Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda B. Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *CoRR* abs/1603.04467 (2016). arXiv:1603.04467 http://arxiv.org/abs/1603.04467

[2] Alan Bawden. 1999. Quasiquotation in Lisp. In *Proceedings of the 1999 ACM SIGPLAN Workshop on Partial Evaluation and Semantics-Based Program Manipulation, San Antonio, Texas, USA, January 22-23, 1999. Technical report BRICS-NS-99-1*. 4–12.

[3] Sarah Bird and Dmytro Dzhulgakov. 2017. Creating an Open and Flexible ecosystem for AI models with ONNX. (2017). Retrieved December 15, 2017 from http://learningsys.org/nips17/assets/slides/ONNX-workshop.pdf

[4] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[5] Josiah L. Carlson. 2013. *Redis in Action*. Manning Publications Co., Greenwich, CT, USA.

[6] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*. 785–794. https://doi.org/10.1145/2939672.2939785

[7] Kristina Chodorow and Michael Dirolf. 2010. *MongoDB: The Definitive Guide* (1st ed.). O'Reilly Media, Inc.

[8] Doug Cutting and Thiruvalluvan M. G. 2016. Apache Avro. (2016). Retrieved Feburary 15, 2017 from http://avro.apache.org version 1.8.1.

[9] Jerome H. Friedman. 2002. Stochastic Gradient Boosting. *Comput. Stat. Data Anal.* 38, 4 (Feb. 2002), 367–378. https://doi.org/10.1016/S0167-9473(01)00065-2

[10] Alexander Grubb and Drew Bagnell. 2012. SpeedBoost: Anytime Prediction with Uniform Near-Optimality. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2012, La Palma, Canary Islands, April 21-23, 2012*. 458–466. http://jmlr.csail.mit.edu/proceedings/papers/v22/grubb12.html

[11] Alex Guazzelli, Wen-Ching Lin, and Tridivesh Jena. 2010. *PMML in Action: Unleashing the Power of Open Standards for Data Mining and Predictive Analytics*. CreateSpace, Paramount, CA.

[12] Jeremy Hermann and Mike Del Balso. 2017. Meet Michelangelo: Uber's Machine Learning Platform. (2017). Retrieved December 19, 2017 from https://eng.uber.com/michelangelo

[13] G. Holmes, A. Donkin, and I.H. Witten. 1994. WEKA: A Machine Learning Workbench. In *Proc Second Australia and New Zealand Conference on Intelligent Information Systems*. Brisbane, Australia.

[14] Hanzhang Hu, Debadeepta Dey, J. Andrew Bagnell, and Martial Hebert. 2017. Anytime Neural Networks via Joint Optimization of Auxiliary Losses. *CoRR* abs/1708.06832 (2017). arXiv:1708.06832 http://arxiv.org/abs/1708.06832

[15] Hanzhang Hu, Alexander Grubb, J. Andrew Bagnell, and Martial Hebert. 2016. Efficient Feature Group Sequencing for Anytime Linear Prediction. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence, UAI 2016, June 25-29, 2016, New York City, NY, USA*. http://auai.org/uai2016/proceedings/papers/86.pdf

[16] Amazon Web Services Inc. 2017. *Algorithms Provided by Amazon SageMaker: Common Parameters*. Retrieved December 19, 2017 from http://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-algo-docker-registry-paths.html

[17] Amazon Web Services Inc. 2017. *Amazon SageMaker: Developer Guide*. Retrieved December 19, 2017 from http://docs.aws.amazon.com/sagemaker/latest/dg/sagemaker-dg.pdf

[18] Amazon Web Services Inc. 2017. *Common Data FormatsâĂŤInference*. Retrieved December 19, 2017 from http://docs.aws.amazon.com/sagemaker/latest/dg/cdf-inference.html

[19] Vineet Khare, Marcio Vinicius dos Santos, and Jonathan Esterhazy. 2017. Converting the Parquet data format to recordIO-wrapped protobuf. (2017). Retrieved December 19, 2017 from https://github.com/awslabs/amazon-sagemaker-examples/blob/master/advanced_functionality/parquet_to_recordio_protobuf/parquet_to_recordio_protobuf.ipynb

[20] Avinash Lakshman and Prashant Malik. 2010. Cassandra: A Decentralized Structured Storage System. *SIGOPS Oper. Syst. Rev.* 44, 2 (April 2010), 35–40. https://doi.org/10.1145/1773912.1773922

[21] John Langford, Lihong Li, and Alexander L. Strehl. 2007. Vowpal Wabbit. (2007). Retrieved Feburary 15, 2017 from http://hunch.net/~vw/

[22] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2010. A Contextual-Bandit Approach to Personalized News Article Recommendation. *CoRR* abs/1003.0146 (2010). arXiv:1003.0146 http://arxiv.org/abs/1003.0146

[23] Xiangrui Meng, Joseph K. Bradley, Burak Yavuz, Evan R. Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, D. B. Tsai, Manish Amde, Sean Owen, Doris Xin, Reynold Xin, Michael J. Franklin, Reza Zadeh, Matei Zaharia, and Ameet Talwalkar. 2016. MLlib: Machine Learning in Apache Spark. *Journal of Machine Learning Research* 17 (2016), 34:1–34:7. http://jmlr.org/papers/v17/15-237.html

[24] Jim Pivarski, Collin Bennett, and Robert L. Grossman. 2016. Deploying Analytics with the Portable Format for Analytics (PFA). In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)*. ACM, New York, NY, USA, 579–588. https://doi.org/10.1145/2939672.2939731

[25] D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-François Crespo, and Dan Dennison. 2015. Hidden Technical Debt in Machine Learning Systems. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*. 2503–2511. http://papers.nips.cc/paper/5656-hidden-technical-debt-in-machine-learning-systems

[26] The H2O.ai team. 2015. *H2O: Scalable Machine Learning*. https://github.com/h2oai/h2o-3 version 3.1.0.99999.

[27] The Scala Team. 2017. *Scala programming language*. EPFL and Lightbend, Inc. http://scala-lang.org/files/archive/spec/2.12/

[28] The TensorFlow team. 2017. *Preparing serving inputs*. Retrieved December 19, 2017 from https://www.tensorflow.org/programmers_guide/saved_model#preparing_serving_inputs Programmer's Guide: Saving and Restoring.

[29] Kenton Varda. 2008. *Protocol Buffers: Google's Data Interchange Format*. Technical Report. Google. http://google-opensource.blogspot.com/2008/07/protocol-buffers-googles-data.html

[30] Wikipedia. 2017. Key-value database — Wikipedia, The Free Encyclopedia. (2017). Retrieved December 15, 2017 from https://en.wikipedia.org/wiki/Key-value_database

[31] Zhixiang Eddie Xu, Matt J. Kusner, Kilian Q. Weinberger, and Minmin Chen. 2013. Cost-Sensitive Tree of Classifiers. In *Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013*. 133–141. http://jmlr.org/proceedings/papers/v28/xu13.html

[32] Xin Yan and Xiao Gang Su. 2009. *Linear Regression Analysis: Theory and Computing*. World Scientific Publishing Co., Inc., River Edge, NJ, USA.

[33] Uwe Zdun, Markus Völter, and Michael Kircher. 2003. Design and Implementation of an Asynchronous Invocation Framework for Web Services. In *Web Services - ICWS-Europe 2003, International Conference ICWS-Europe 2003, Erfurt, Germany, September 23-24, 2003, Proceedings*. 64–78. https://doi.org/10.1007/978-3-540-39872-1_6