Faster Evaluation of Labor-Intensive Features

Michael R. Anderson
University of Michigan
mrander@umich.edu

Feature engineering—developing a set of values that effectively describe raw data for a machine learning task—can be a lengthy, repetitive process. Further, good features for one learning task may not be suitable for another, so creating effective features can largely be a process of trial-and-error. Difficulties aside, however, the ultimate success of a trained system depends on the ability of the features to accurately represent the task-specific variations within the raw data.

Developing features can be highly iterative, with significant engineer downtime during the execution of the feature functions over the raw dataset. Reducing this downtime can improve both the engineer's productivity and the trained system's overall quality. We model the feature engineering process as follows:

1. The feature engineer writes or modifies a set feature functions that process a raw data input and emit a vector of numeric values.
2. The feature functions are applied to the raw data using a bulk data processing system, such as MapReduce, producing a training set of feature vectors.
3. The training set is provided to a machine learning system, which is trained to produce a learned artifact.
4. The engineer evaluates the quality of the learned artifact using standard evaluation metrics. If the artifact is judged not good enough, she begins again at Step 1.

While most traditional machine learning research focuses on improving the learning system itself (Step 3), our research focuses on the efficient development and execution of feature functions over very large raw datasets (Steps 1 and 2).

Standard feature sets exist for many learning tasks, such as token counts for text-based tasks and SIFT for computer vision tasks. Commodity features like these are often the first attempt at developing a feature set for a learning system. However, these features are by necessity general: they cannot take advantage of nuances in a specific raw dataset or learning task. Using commodity features as a starting point, a feature engineer can develop additional features tailored to the particular dataset by applying domain expertise [1, 3].

7th Biennial Conference on Innovative Data Systems Research (CIDR '15) January 4-7, 2015, Asilomar, California, USA.

We have implemented a system that saves considerable time in the above development cycle by reducing the time cost of generating the feature vectors for a suitable training set (Step 2) [2]. Like giving a developer a faster compiler, these savings allow the feature engineer to iterate on feature code quickly; tasks can be completed faster or with a higher quality outcome than under standard development processes.

Our system leverages the observation that in a large corpus of raw data, such as a web crawl, a large amount of the data items are redundant or irrelevant to a given learning task. By quickly finding potential inputs that *are* relevant and not yet redundant, our system can generate a training set that is much smaller, yet just as effective as one generated by using the entire raw dataset. The reduction in training set size directly translates into a reduction in feature extraction processing time (Step 2) because far less of the dataset is processed. (Small training sets are also the goal of active learning [4], but those methods typically require generating the features for all raw data—exactly what we wish to avoid.)

In a one-time initialization procedure, the raw dataset is clustered using a general clustering method suitable for the data type and an index data structure is built over these clusters. Using online learning techniques, our system builds a mapping between the clusters and the internal state of the learning system during the feature extraction process. It can quickly identify and exploit the raw data clusters likely to contain items that will produce feature vectors useful to the training set. The overhead of this input selection process is negligible for all but the simplest feature functions, and our experiments show that our system can generate a good training set from three to ten times faster than standard methods for a variety of learning tasks.

The features used for a machine learning task are essential for a successful trained system, and engineering great features is a difficult task. Our research aims to remove some of the tedium and unproductive thumb twiddling inherent in the current practice of feature engineering by providing tools to find the right features for the right data, allowing the feature engineer to quickly build effective machine learning systems.

REFERENCES

[1] M. Anderson, D. Antenucci, V. Bittorf, M. Burgess, M. Cafarella, A. Kumar, F. Niu, Y. Park, C. Ré, and C. Zhang. Brainwash: A data system for feature engineering. In *CIDR*, 2013.

[2] M. R. Anderson, M. Cafarella, Y. Jiang, G. Wang, and B. Zhang. An integrated development environment for faster feature engineering. In *VLDB*, 2014.

[3] P. Domingos. A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78, 2012.

[4] B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.