TITLE

Alyssa Morrow
akmorrow@berkeley.edu

Devin Petersohn
devin.petersohn@berkeley.edu

Anthony Joseph
adj@berkeley.edu

Nir Yosef
nir.yosef@berkeley.edu

ABSTRACT

Current pipelines for training machine learning models on genomic and epigenetic datasets lack resources to scale to current datasets sizes. Furthermore, pipelines for testing and evaluating these machine learning models are segmented, as they require users to separately test pipelines and evaluate their methods using disjunct software. In this paper we survey current workflows for training, testing and evaluating genomic machine learning models and provide an alternative solution to connecting the dots between these processes. More specifically, present a complete fast and interactive pipeline that allows users to save prebuilt models and evaluate these models in a visual environment with low latency.

Keywords

Genomics; Model Serving; Transcription Factors

1. INTRODUCTION

Regulatory genomics is a field of research that works to connect the function of DNA sequence, DNA interacting proteins and cell environment with gene expression. Understanding and predicting regulatory function is important, as it allows clinicians to predict the effect of an individual's genetic variants on their health and well being. Currently, there exist many different computational methods for prediction of regulatory function, including convolutional neural networks and SVM classification problems (Alipanahi et. al. 2014, Kelley et. al. 2016). Datasets used to predict regulatory function are massive, reaching 15 TB in the ENCODE consortium (cite). However, these datasets are noisy, and thus analysis of regulatory predictions often require a human in the loop to visually interpret predictions against ground truth datasets produced in datasets such as ENCODE.

Current pipelines for training and analyzing regulatory genomic models include a segmented group of steps that are exclusive and require preprocessing between each stage. For example, a state of the art pipeline involves a local algorithm, such as a kernel method or neural network, built for a single machine. Because these methods are not built to scale past a single machine, users must first subselect regions of the genome to train on. A user would then subselect certain areas of the genome, and predict on these regions. Once the user has predicted on these regions, the user would then load these regions into a single node visualization tool, such as IGV, and repeat this preprocessing step to validate their model. A demonstration can be shown in Figure!1.

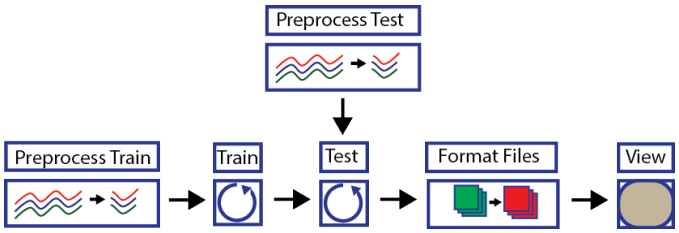

Figure 1: Stawman pipeline for training and evaluating model.

Here, we provide an end-to-end pipeline that efficiently computes and predicts on regulatory genomics while maintaining seamless integration between processing steps. In our previous work, we use a scalable and parallel algorithm to efficiently compute protein binding sites without the use of specialized hardware or $O(n^2)$ complexity (cite). Here, we utilize this algorithm to predict from regulatory datasets, and provide a full analysis pipeline to analyze results. Our pipeline allows users to interactively predict on regions of the genome, removing file preprocessing steps in the state of the art analysis pipelines. We compare our pipeline to the current state of the art approaches intended for a local machine, and then analyze pipeline latency at scale.

2. BACKGROUND

ALYSSA TODO Current algorithms (neural nets and SVM) Current tools for predicting on regulatory genomic datasets include neural networks and SVM classification. These methods train on small portions of the genome. For example, DeepBind subselects 10,000 sequences to predict protein binding sites. Similarily, Kernel SVM uses 10,000 sequence to learn a classifier. Both tools are intended to run on a single machine.

PLDI '13 June 16–19, 2013, Seattle, WA, USA

DOI: 10.475/123_4

Vis tools (IGV) The Integrated Genomics Viewer (IGV) is a desktop application that allows visualization of a wide variety of genomic datasets. IGV is used by clinicians and researchers to verify anomalies in their data and verify predictions to original raw genomic datasets. IGV does not support integrated visualization of predictive models, so users attempting to perform visual analysis of models using heterogeneous datasets must predict outside of the visualization tool.

3. ARCHITECTURE

In this section, we outline our design for an end-to-end machine learning pipeline on epigenetic datasets. Our implementation builds upon previously built distributed genomics tools such as ADAM, Mango and Endive, which are described below. Finally, we will demonstrate how these tools can be connected together, with our new machine learning tool, to create an end-to-end pipeline for machine learning tasks.

ADAM is a framework for processing genomic data, and is built to run in the Spark ecosystem. ADAM uses distributed abstractions in Spark to batch process raw genomic datasets in parallel. More importantly, ADAM provides standardized schemas intended for various types of genomic datasets, presented in a narrow waist design that can be easily accessible by external applications. A figure of schema design is shown in Figure X. Important schemas that are used for epigenetic datasets include Alignment Record, Features, and Variant schemas. These schemas are demonstrated in Figure!??.

Mango Mango is a distributed genome visualization browser that selectively materializes and organizes genomic data to provide fast in-memory queries. Mango materializes data from persistent storage as the user requests different regions of the genome. This data is efficiently partitioned and organized in memory using a specialized RDD structure built upon interval trees. This interval based organizational structure supports ad hoc queries, filters, and joins across multiple files, enabling exploratory interaction with genomic data. Mango is built on top of Spark and ADAM. A demonstration of Mango architecture can be seen in Figure X. Mango is implemented in three layers, namely, the cluster, server and client. In the cluster layer, Mango is built on ADAM and Spark, allowing users to perform and visualize fine grained queries on large genomic files. The server layer reduces and samples queries so that users can visualize their queries in an efficient and interpretable manner.

Our machine learning tool, called Endive, provides core abstract stages for building end-to-end machine learning pipelines to train and evaluate on genomic datasets. These stages include data preprocessing, featurization, and various solvers apt for genomic workloads. These stages in a machine learning pipeline can be mixed and matched to create novel pipelines tailored to a genomic prediction task of interest. Preprocessing in Endive involves taking raw genomic data and parsing it into a vector of numeric values which can be directly read by a machine learning model. These preprocessing steps include formatting datasets as well as joining together heterogeneous datasets into one record based on coordinate location in the genome. Featurization steps in Endive involve taking preprocessed datasets created in the preprocessing stage and transforming these records to explicit features that can be trained and tested on. Table 1 contains examples of these featurization steps. These computed features can then be fed into a choice of solvers developed in Keystone ML to train these machine learning models (Sparks, 2015).

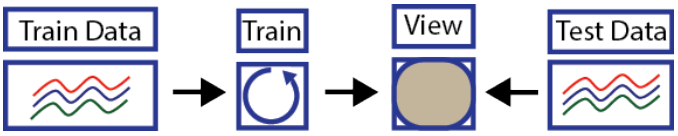

Figure 2: Pipeline overview. We eliminate preprocessing and file formatting steps.

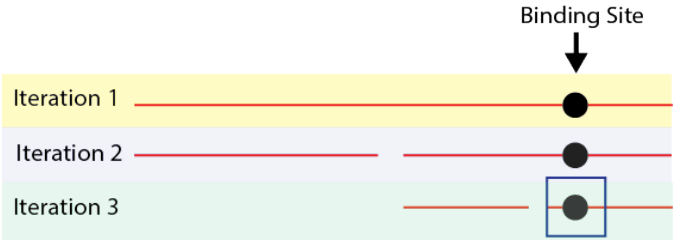

Figure 3:

Machine Learning Pipeline for Epigenetic Workloads For the entirety of this paper, we will focus on using kernel approximations described in Table 1 to predict protein binding sites for an example pipeline.

3.1 Interactive Prediction of Binding Sites

Initially, direct prediction of binding sites was not in interactive time (500ms) (cite). To reduce latency, we took advantage that regulatory regions are fairly sparce, touching only 0.01

4. EXPERIMENTS

For preliminary results, we train a model predicting transcription factor binding sites using the kernel approximation featurization method described in Table 1 from ENCODE datasets using DNA sequence. This model was solved using Keystone MLâĂŹs Weighted Least Squares estimator, which accounts for large class imbalance of positive binding sites across the genome. This model was trained on 200,000,000 regions of the genome and generated a 4 MB model to be tested on new datasets. This model was loaded into the Mango genome browser and was used to interactively predict on DNA reference sequences as the user scrolled to new regions of the browser.

4.1 Local Comparison

For a straw man comparison, we took the state of the art pipeline used to train and analyze protein binding sites on a local computer. We trained a model on 15,000 DNA sequences to predict the location of binding sites. These models were trained and tested on the EGR1 transcription factor. The strawman pipeline includes training a model with DeepBind, a neural net implementation to predict binding sites from DNA sequences [cite]. To visualize results, we used IGV, a local genome browser.

Our pipeline trains a model on 15,000 sequences and uses Mango to analyze results. Because the models were too

expensive to train locally, we train both straw man and our models on one machine with 24 Xeon processors, and 256 GB of ram and 1 Nvidia Tesla K20c GPU.

DEVIN RESULTS

4.2 Scalable Pipeline for Model Analysis

For our second experiment, we trained on the full genome, or 200,000,000 sequences to predict binding sites for EGR1.

ALYSSA TODO

5. CONCLUSION

DEVIN

6. ADDITIONAL AUTHORS

7. REFERENCES

7.1 References

Generated by bibtex from your .bib file. Run latex, then bibtex, then latex twice (to resolve references) to create the .bbl file. Insert that .bbl file into the .tex source file and comment out the command `\thebibliography`.