**Big Data and Machine Learning**

**Technology in IceCube**

**Introduction**

The recent development in big data and machine learning technology has dramatically

changed how business performs large scale computing and AI analytics. The use of ML and

AI has been accelerating in data-driven organizations with the help of tools such as Apache

Hadoop, Apache Spark and many popular Python libraries such as numpy, pandas,

Scikit-Learn, PySpark, TensorFlow (Google) and PyTorch (Facebook). The big data and ML

tools not only can help physicists gain insights and solve complex cosmic-ray problems. But

dramatically reduce the burden in software engineering work to build up the data pipeline,

machine learning algorithm and results evaluation. This document will discuss a few

mainstream big data and ML tools and their potential usage in IceCube.

**Data Storage and Loading**

**Numpy & Pandas**

Those are python libraries for data processing and scientific computation. For data with a size that can be read in the memory of your machine, probably they are the best tools to handle data. Besides, the numpy array and pandas dataframe are the foundation of other Python machine learning libraries such Scikit-Learn, TensorFlow and PyTorch. The have simple APIs to read csv, json, excel and HDF5 format. Extra work would be needed to deal with IceCube’s I3 binary data files.

For data of size > 100GB that are distributed on a network, big data software such as Hadoop, Spark or TensorFlow could be used process and analyse the data, both of which are open-sourced.

**Apache Hadoop**

Hadoop is pioneer in big data industry as it first introduced the HDFS storage system and MapReduce programming model. However, it’s written in Java and may be suitable to large business organizations. It has too many low-level APIs so small groups may find it too costly to operate their own Hadoop system.

**Apache Spark**

Spark started as a child project of Hadoop but developed into a unified platform for parallel data processing on clusters. It’s designed to support a wide range of data analytics tasks, ranging from simple data loading and SQL queries to ML and streaming computation with simple APIs in multiple languages (Python, Java, Scala …). It also has a rich ML library. With Spark’s high-level APIs, it’s easy to have data loading, processing, cleansing and doing machine learning integrated. Spark is very popular in tech companies such as Netflix and Uber, as well as research institutions like NASA and CERN. [1]

**TensorFlow**

TensorFlow’s data API (with Keras) provides users efficient ways to read from text files, binary files with fixed-size records, and binary files that use TensorFlow’s TFRecord format (based on Protocol Buffers), which supports records of varying sizes. [2]

**Data Transformation**

Big data analytics and machine learning algorithms are picky about the cleanse and accuracy of the data. Data cleaning is only the first step to prepare the data for machine learning algorithms. A few data transformation may take place before feed the raw data to any machine learning algorithm. For example, the numeric features may need to be scaled, categorical features may need to be encoded, new features may be added, PCA (principle component analysis) may be performed to reduce dimensions, etc. The data samples are then shuffled and split into train, validation and test sets. Scikit-Learn, TensorFlow and Spark have their own data pipeline tools to standardize the transform to make sure the data transform is consistent among different runs.

**Scikit-Learn**

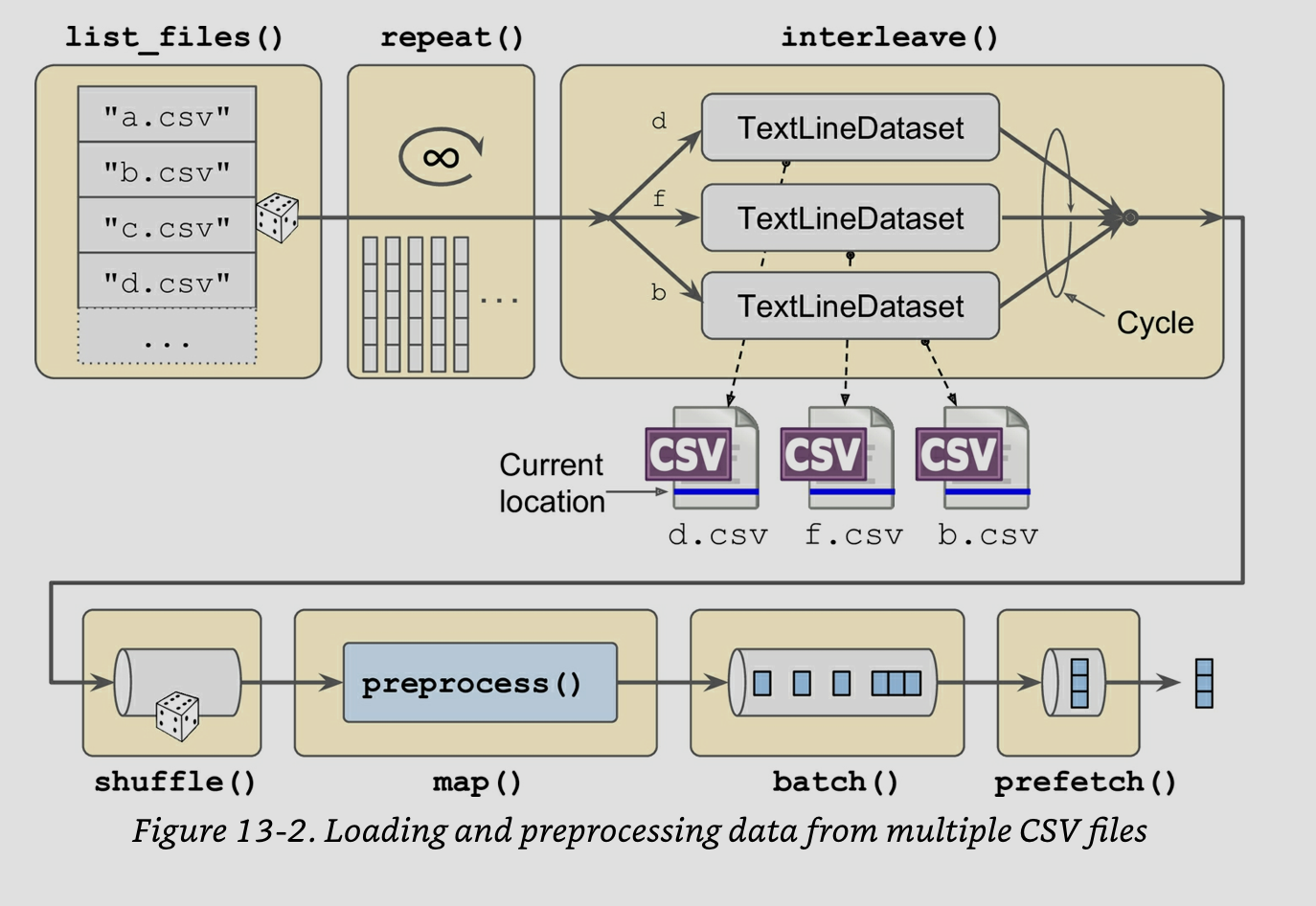
Scikit-Learn has some handy pipeline and encoding functions to facilitate the data transformation. Moreover, the users can define their own custom transform class in Python.

**TensorFlow**

TensorFlow data API has many data manipulation functions. Currently the Kreas preprocessing layers is still under development. An example of loading and preprocessing data from multiple CSV files:

**Apache Spark**

Spark dataframe allows the users to manipulate the data with functions mimicking SQL commands.



**Machine Learning Models**

Machine learning algorithms have a broad spectrum from simple linear regression to deep neural networks. Luckily the Scikit-Learn package has rich selection of model APIs. We turn to TensorFlow (or PyTorch) for deep learning.

**Scikit-Learn**

Scikit-Learn has easy API for such models: linear regression, gradient descent, polynomial regression, ridge regression, lasso regression, logistic regression, SVM, decision trees, ensemble learning and random forest, boosting, dimension reduction techniques such as PCA and some unsupervised learning techniques such as KMeans.

Scikit-Learn also provides grid or random search functions to automatically select the best performing hyperparameters to fit the models for users, which would shorten the model tuning process significantly.

In cosmic-ray physics, we often need to estimate the energy or categorize the type of a particle. The MC simulated data has both features and the labels of the particle. We could use the MC data to train the various regressors or classifiers in Scikit-Learn and use the model to “predict” the energy or type of the experiment observed particles. Scikit-Learn’s resampling module should also help to deal with imbalanced datasets that often occur in cosmic-ray problems.

**TensorFlow**

TensorFlow is probably the most important data processing and machine learning framework today known as its capability to perform deep learning. Currently TensorFlow 2.0 is fully integrated with the Keras APIs. The users can choose the high-level API to build the neural networks by defining layers (input, hidden and output layers), optimizers, loss functions, initialization methods, activation function and etc. Alternatively, the users can use low-level APIs to define their own customized components. The users can also leverage on Scikit-Learn’s grid search method to find the optimal hyperparameters automatically. Given IceCube’s abundant cosmic-ray data in certain energy range, deep neural networks could be used to solve some physics problems. For example, we can build networks to reconstruct the path or the incoming particles. In the problem, each of the 6000 DOM’s response could be a feature and the labels are the particles’ track, energy and type. Such a network can have an input layer with shape [6000, 1], any number of hidden layers and an output layer with 6 neurons (4 space parameters, one energy estimator and the mass).

TensorFlow also makes it easy to perform CNN (Convolutional Neural Networks) and RNN (Recurrent Neural Networks). The former would treat IceCube firing as images and do classification while the latter would best be used in time series forecasting or natural language processing.

**Spark MLlib**

MLlib is the machine learning component to Apache Spark parallel to Scikit-Learn or TensorFlow. Its ALS model is popular in building recommendation systems. I feel it’s a little slow compared to other frameworks. However, if you have to process petabyte of data all at once or handle streaming data, Apache Spark might be the right choice. [3]

**Models Evaluation Methodology**

Machine learning is not completely studied in all scenarios. Sometimes we can only judge a model by evaluating its performance in certain cases. Luckily both Scikit-Learn and TensorFlow have a rich selection of functions to do this.

**Scikit-Learn**

Scikit-Learn can do cross validation. In binary classification, it can output the precision-recall and ROC curves. It can also output the confusion matrix for classification problems. For instance, suppose we have a model that can distinguish neutrinos from other particles. We simulated 100 neutrinos and 1000 other particles to train the model and get the confusion matrix:

90 10

50 950

The matrix means out of the 100 true neutrinos, the model selected 90 so recall = 90 / 100 = 90%. Among the 90 + 50 = 140 selected neutrinos, only 90 are true so precision = 90 / 140 = 64.3%.

For regression models MSE (mean squared error) is often used.

**TensorFlow**

TensorFlow has similar evaluation methods.

**Database Technology**

Given the amount and complex of IceCube data, relational database such as PostgreSQL may not be relevant. However, as an import part of big data technology it’s worth mentioning. Besides, many visualization tools prefer fetching data from a database.

KDB+q is the most advanced time series data nowadays. The license is quite expensive but the 32bit version is free. The only use case is to deal with millions of rows of time series data. It’s extremely fast.

**Data Visualization**

Tableau is probably the most popular data visualization tools for large companies though not cheap. My second choice would be Plotly-Dash which gives users flexibility with Python development. Metabase can produce beautiful dashboards without doing any coding or SQL query. Superset was originally developed by AirBnb. It has many features if you’re strong in SQL. Attached is an example of Metabase dashboard:



**Cloud and Microservices**

AWS & GCP could provide scalable computing resources. Some computation could be moved to cloud to reduce the cost. Docker & Kubernetes have been gaining popularity in industry. IceCube institutions are distributed all over the world. Such technology could make life easier for developers.

**Summary**



In recent years, large technology conglomerates have put enormous effort to develop open-sourced big data and machine learning algorithms. Tools such as Numpy, Pandas, Scikit-Learn, TensorFlow and Apache Spark have become industry norm. The procedures of performing machine learning have also been standardized. The standard procedures are [4]

1. Look at the big picture.
2. Get the data.
3. Discover and visualize the data to gain insights.
4. Prepare the data for ML algorithms.
5. Select models to train it.
6. Fine tune the models.
7. Present your solution.
8. Launch, monitor and maintain your system.

A diagram is to show the process at the end of the document.

Each of the steps can be developed with the industry’s standard methods. Adopting the mainstream ML methods would help IceCube achieve:

1. Be able to use the latest machine learning results from academia.
2. Out-source low-level infrastructure development. Since they’re generally maintained by big tech firms such as Google, it’s likely to be more reliable than if developed in-house.
3. Be able to share and cross-check ML code easily among different IceCube offices. Scientific results are thus easier to be replicated.
4. Use standard model evaluation methodology could make it easier to present results to other scholars who do not major in Physics.



[1] Chapter 1 & 2, Spark the Definitive Guide by Bill Chambers & Matei Zaharia

[2] Chapter 13, Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow by Aurelien Geron

[3] Chapter 24, Spark the Definitive Guide by Bill Chambers & Matei Zaharia

[4] Chapter 2, Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow by Aurelien Geron