File Formats

File Formats

- This video introduces the most widely used file formats for ML, grouping them into known classes of well-known file format types: columnar, tabular, nested, array-based, and hierarchical.

Columnar Data File Formats

- Columnar file formats are designed for use on distributed file systems (HDFS, HopsFS) and object stores (S3, GCS, ADL) where workers can read the different files in parallel.
- ***File formats*: .parquet, .orc, .petastorm.**
- *Feature Engineering*: PySpark, Beam, Flink.
- *Training*:.petastorm has native readers in TensorFlow and PyTorch;
  .orc, .parquet have native readers in Spark;
  JDBC/Hive sources supported by Spark
-

Tabular

- ***File formats*: .csv, .xslx**
- *Feature Engineering*: Pandas, Scikit-Learn, PySpark, Beam, and lots more
- *Training*: .csv has native readers in TensorFlow, PyTorch, Scikit-Learn, Spark

Nested File Formats

- ***File formats*: .tfrecords, .json, .xml, .avro**
- *Feature Engineering*: Pandas, Scikit-Learn, PySpark, Beam, and lots more
- *Training*: .tfrecords is the native file format for TensorFlow;
  .json has native readers in TensorFlow, PyTorch, Scikit-Learn, Spark
  .avro files can be used as training data in TensorFlow with LinkedIn's library.

Array-Based Formats

- Numpy is an abbreviation of Numerical Python and it is a massively popular library for scientific computing and data analysis. Through support for vectorization, Numpy (.npy) is also a high performance file format. A Numpy array is a densely packed array with elements of the same type. The file format, .npy, is a binary file format that stores a single NumPy array (including nested record arrays and object arrays).
- ***File formats*: .npy**
- *Feature Engineering*: PyTorch, Numpy, Scikit-Learn, TensorFlow;
- *Training*: .npy has native readers in PyTorch, TensorFlow, Scikit-Learn.

Hierarchical Data Formats

- HDF5 (.h5 or .hdf5) and NetCDF (.nc) are popular hierarchical data file formats (HDF) that are designed to support large, heterogeneous, and complex datasets.
- In particular, HDF formats are suitable for high dimensional data that does not map well to columnar formats like parquet (although petastorm is both columnar and supports high dimensional data). Lots of medical device data is stored in HDF files or related file formats, such as BAM, VCF for genomic data.
- Internally, HDF5 and NetCDF store data in a compressed layout. NetCDF is popular in domains such as climate science and astronomy. HDF5 is popular in domains such as GIS systems.
- They are not splittable, so not suitable for distributed processing (with engines like Spark).

Choosing file formats

- Some things to consider when choosing the format are:
- **The structure of your data**: Some formats accept nested data such as JSON, Avro or Parquet and others do not. Even, the ones that do, may not be highly optimized for it. Avro is the most efficient format for nested data, I recommend not to use Parquet nested types because they are very inefficient. Process nested JSON is also very CPU intensive. In general, it is recommended to flat the data when ingesting it.
- **Performance**: Some formats such as Avro and Parquet perform better than other such JSON. Even between Avro and Parquet for different use cases one will be better than others. For example, since Parquet is a column based format it is great to query your data lake using SQL whereas Avro is better for ETL row level transformation.
- **Easy to read**: Consider if you need people to read the data or not. JSON or CSV are text formats and are human readable whereas more performant formats such parquet or Avro are binary.
- **Compression**: Some formats offer higher compression rates than others.
- **Schema evolution**: Adding or removing fields is far more complicated in a data lake than in a database. Some formats like Avro or Parquet provide some degree of schema evolution which allows you to change the data schema and still query the data. Tools such Delta Lake format provide even better tools to deal with changes in Schemas.
- **Compatibility**: JSON or CSV are widely adopted and compatible with almost any tool while more performant options have less integration points.

Conclusion

- CSV and JSON are easy to use, but make it too slow to be used to query the data lake.
- **ORC and Parquet** are widely used in the Hadoop ecosystem to **query data** whereas **Avro** is also used outside of Hadoop, especially together with Kafka for ingestion, it is very good for **row level ETL** processing.
- Row oriented formats have better schema evolution capabilities