Scalable Machine Learning with Dask

Tom Augspurger
Data Scientist
Hi! I’m Tom

- I work at Anaconda on dask, pandas, & ML things
Machine Learning Workflow

It’s not just `.fit(X, y)`
Machine Learning Workflow

- Understanding the problem, objectives
- Reading from data sources
- Exploratory analysis
- Data cleaning
- Modeling
- Deployment and reporting
The Numeric Python Ecosystem

Jake Vanderplas PyCon 2017 Keynote
Machine Learning Workflow

A rich ecosystem of tools

*But they don’t scale well*
Dask
Parallelizing the Numeric Python Ecosystem
Dask

High Level: Parallel Pandas, NumPy, Scikit-Learn

Low Level: High performance task scheduling
import dask.dataframe as dd

df = dd.read_csv('s3://abc/*.csv')
df.groupby(df.name).value.mean()
Dask

High-Level: Scalable NumPy Arrays

```python
import dask.array as da

x = da.random.random(...)
y = x.dot(x.T) - x.mean(axis=0)
```
Dask

Low-Level: Scalable, Fine-Grained Task Scheduling
Dask

- Parallelizes libraries like NumPy, Pandas, and Scikit-Learn
- Adapts to custom algorithms with a flexible task scheduler
- Scales from a laptop to thousands of computers
- Integrates easily, Pure Python built from standard technology
Scaling Pains

Model

Data
Scaling Pains

Model

Scikit-Learn

RAM

Data
Scaling Pains

Model

Distributed
Scikit-Learn

Scikit-
Learn

RAM

Data
Scaling Pains

- Sampling
- Out-of-core algos
- Scalable algos
- Other libraries
Distributed Scikit-Learn
Distributed Scikit-Learn

Large models, smaller datasets

• Use dask to distribute computation on a cluster
Distributed Scikit-Learn

Single-Machine Parallelism with Scikit-Learn

```python
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(n_estimators=200, n_jobs=-1)

clf.fit(X, y)
```
Distributed Scikit-Learn

Multi-Machine Parallelism with Dask

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.externals import joblib
import dask_ml.joblib

clf = RandomForestClassifier(n_estimators=200, n_jobs=-1)

with joblib.parallel_backend("dask", scatter=[X, y]):
    clf.fit(X, y)
```
Distributed Scikit-Learn

Caveats

- Data has to fit in RAM
- Data shipped to each worker
  - Each parallel task should be expensive
  - There should be many parallel tasks
Scalable Algorithms
When your dataset is larger than RAM
First: Do you need all the data?

- Sampling may be OK
- **Plotting Learning Curves** from scikit-learn docs
from dask_ml.wrappers import ParallelPostFit
import dask.dataframe as dd

clf = ParallelPostFit(SVC())
clf.fit(X_small, y_small)

X_large = dd.read_csv("s3://abc/*.parq")
y_large = clf.predict(X_large)

- Train on subset
- Predict for large dataset, in parallel
Scalable Estimators

When the training dataset is larger than RAM

• Scikit-Learn wasn’t designed for distributed datasets
• Dask-ML implements scalable variants of some estimators
• Works well with Dask DataFrames & Arrays
Scalable, Parallel Algorithms

Spectral Clustering Comparison
Scalable, Parallel Algorithms

Some Notable Estimators

- Distributed GLM
  - LogisticRegression, LinearRegression, ...
- Clustering
  - KMeans(init='k-means||'), SpectralClustering, ...
- Preprocessing
  - QuantileTransformer, RobustScalar, ...
- Dimensionality Reduction
  - PCA, TruncatedSVD
- ...
Familiar
Works well with existing libraries
Familiar

• Dask-ML estimators are Scikit-Learn estimators
• Dask-ML pipelines are Scikit-Learn Pipelines
>>> from sklearn.pipeline import make_pipeline
>>> from sklearn.preprocessing import FunctionTransformer

>>> pipe = make_pipeline(
...     ColumnSelector(columns),
...     HourExtractor(['Trip_Pickup_DateTime']),
...     FunctionTransformer(payment_lowerer, validate=False),
...     Categorizer(categories),
...     DummyEncoder(),
...     StandardScaler(scale),
...     LogisticRegression(),
... )
Familiar

```python
>>> from sklearn.pipeline import make_pipeline
>>> from sklearn.preprocessing import FunctionTransformer

>>> pipe = make_pipeline(  
...     ColumnSelector(columns),  
...     HourExtractor(['Trip_Pickup_DateTime']),  
...     FunctionTransformer(payment_lowerer, validate=False),  
...     Categorizer(categories),  
...     DummyEncoder(),  
...     StandardScaler(scale),  
...     LogisticRegression(),  
... )
```

Scikit-Learn objects

Custom transformers

Dask-ML estimators

Full Example: [https://git.io/vAi7C](https://git.io/vAi7C)
Distributed Systems
Integrate with XGBoost and Tensorflow
Distributed System
Peer with systems like XGBoost or Tensorflow

```python
>>> import dask_ml.xgboost as xgb
>>> df = dd.read_csv("trips*.csv")
>>> y = df['Tip_Amt'] > 0
>>> X = df[columns]
>>> booster = xgb.train(
...    client, params, X, y
...)
```
Dask & Dask-ML

- Parallelizes libraries like NumPy, Pandas, and Scikit-Learn
- Scales from a laptop to thousands of computers
- Familiar API and in-memory computation
- [https://dask.pydata.org](https://dask.pydata.org)
Questions?