Dask-ML Benchmarks Documentation

Release 1.0.0

Tom Augspurger

May 17, 2018

Contents:

CHAPTER 1

Parallelizing Predicion

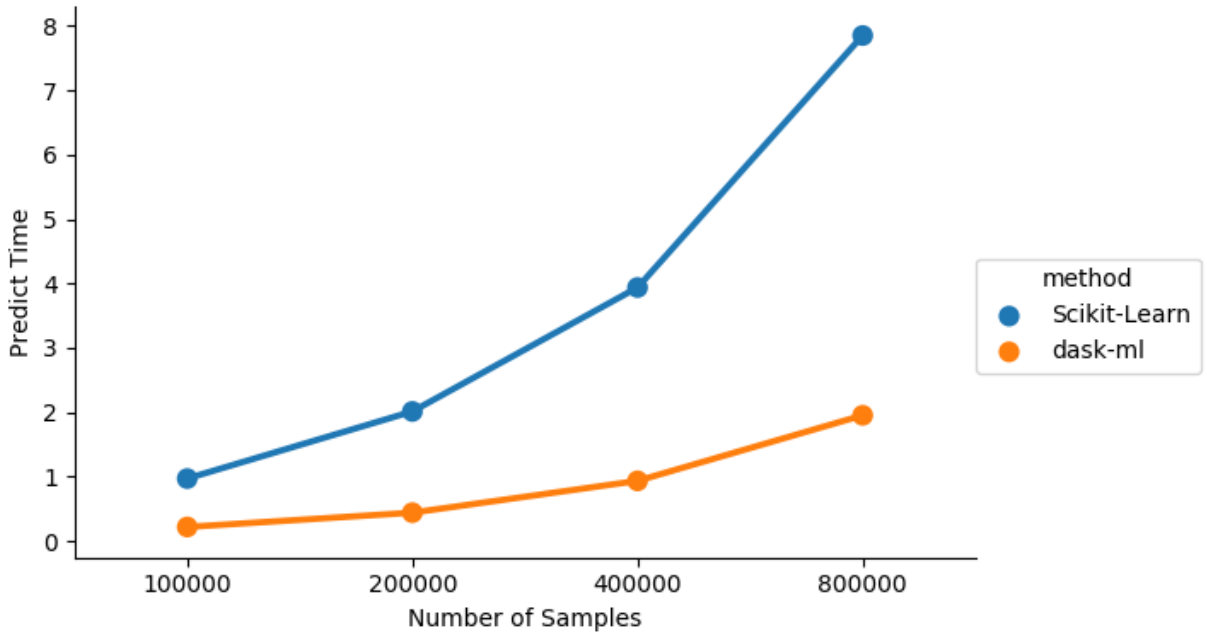
This example demonstrates `dask_ml.wrappers.ParallelPostFit`. A `sklearn.svm.SVC` is fit on a small dataset that easily fits in memory.

After training, we predict for successively larger datasets. We compare

- The serial prediction time using the regular SVC.predict method
- The parallel prediction time using `dask_ml.warppers.ParallelPostFit.predict()`

We see that the parallel version is faster, especially for larger datasets. Additionally, the parallel version from ParallelPostFit scales out to larger than memory datasets.

While only predict is demonstrated here, wrappers.ParallelPostFit is equally useful for predict_proba and transform.




```
from timeit import default_timer as tic

import pandas as pd
import seaborn as sns
import sklearn.datasets
from sklearn.svm import SVC

import dask_ml.datasets
from dask_ml.wrappers import ParallelPostFit

X, y = sklearn.datasets.make_classification(n_samples=1000)
clf = ParallelPostFit(SVC(gamma='scale'))
clf.fit(X, y)


Ns = [100_000, 200_000, 400_000, 800_000]
timings = []


for n in Ns:
    X, y = dask_ml.datasets.make_classification(n_samples=n,
                                                random_state=n,
                                                chunks=n // 20)
    t1 = tic()
    # Serial scikit-learn version
    clf.estimator.predict(X)
    timings.append(('Scikit-Learn', n, tic() - t1))

    t1 = tic()
    # Parallelized scikit-learn version
    clf.predict(X).compute()
    timings.append(('dask-ml', n, tic() - t1))
```

(continues on next page)

(continued from previous page)

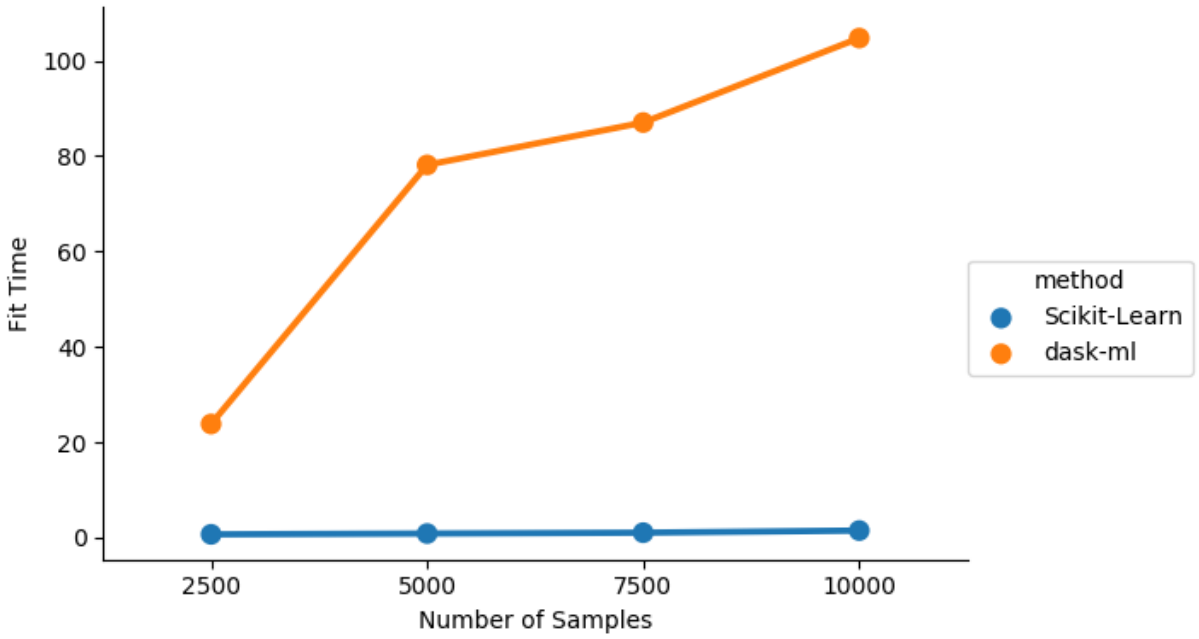
```
df = pd.DataFrame(timings,
                  columns=['method', 'Number of Samples', 'Predict Time'])
ax = sns.factorplot(x='Number of Samples', y='Predict Time', hue='method',
                    data=df, aspect=1.5)
```

Total running time of the script: (0 minutes 22.372 seconds)

CHAPTER 2

Logistic Regression Example

Comparison of scaling.




```
from dask_ml.datasets import make_classification
import pandas as pd

from timeit import default_timer as tic
import sklearn.linear_model
import dask_ml.linear_model
import seaborn as sns
```

(continues on next page)

(continued from previous page)

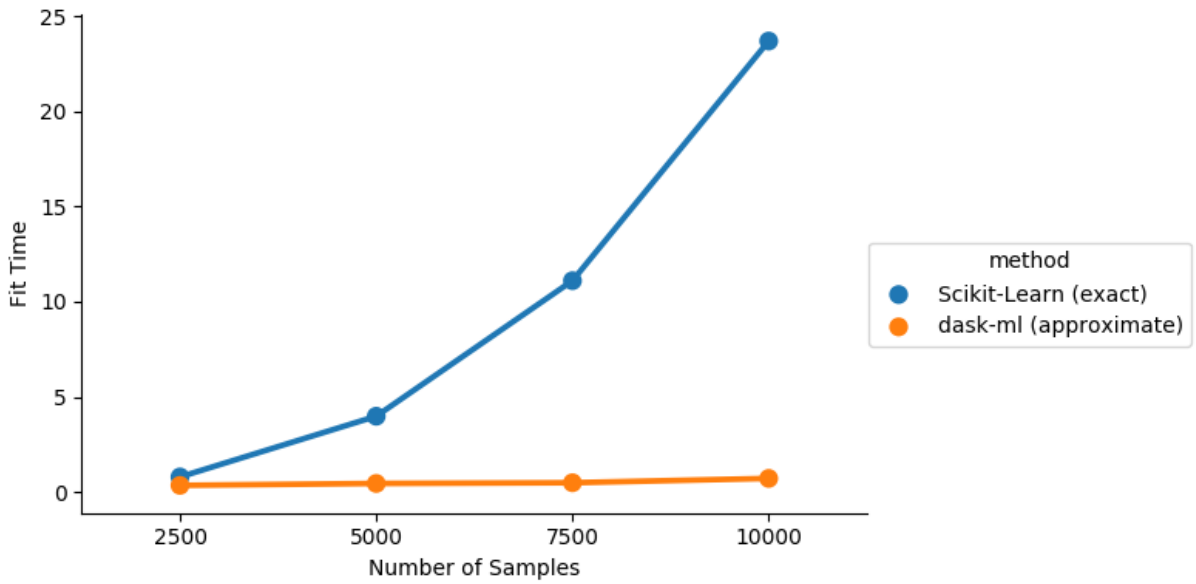
```
Ns = [2500, 5000, 7500, 10000]

timings = []

for n in Ns:
    X, y = make_classification(n_samples=n, n_features=1_000, random_state=n,
                               chunks=n // 20)
    t1 = tic()
    sklearn.linear_model.LogisticRegression().fit(X, y)
    timings.append(('Scikit-Learn', n, tic() - t1))
    t1 = tic()
    dask_ml.linear_model.LogisticRegression().fit(X, y)
    timings.append(('dask-ml', n, tic() - t1))


df = pd.DataFrame(timings, columns=['method', 'Number of Samples', 'Fit Time'])
sns.factorplot(x='Number of Samples', y='Fit Time', hue='method',
               data=df, aspect=1.5)
```

Total running time of the script: (5 minutes 0.900 seconds)

CHAPTER 3

Spectral Clustering Example

This example shows how dask-ml's `SpectralClustering` scales with the number of samples, compared to scikit-learn's implementation. The dask version uses an approximation to the affinity matrix, which avoids an expensive computation at the cost of some approximation error.




```
from sklearn.datasets import make_circles
from sklearn.utils import shuffle
import pandas as pd

from timeit import default_timer as tic
import sklearn.cluster
import dask_ml.cluster
import seaborn as sns
```

(continues on next page)

(continued from previous page)

```
Ns = [2500, 5000, 7500, 10000]
X, y = make_circles(n_samples=10_000, noise=0.05, random_state=0, factor=0.5)
X, y = shuffle(X, y)

timings = []
for n in Ns:
    X, y = make_circles(n_samples=n, random_state=n, noise=0.5, factor=0.5)
    t1 = tic()
    sklearn.cluster.SpectralClustering(n_clusters=2).fit(X)
    timings.append(('Scikit-Learn (exact)', n, tic() - t1))
    t1 = tic()
    dask_ml.cluster.SpectralClustering(n_clusters=2, n_components=100).fit(X)
    timings.append(('dask-ml (approximate)', n, tic() - t1))


df = pd.DataFrame(timings, columns=['method', 'Number of Samples', 'Fit Time'])
sns.factorplot(x='Number of Samples', y='Fit Time', hue='method',
               data=df, aspect=1.5)
```

Total running time of the script: (0 minutes 42.835 seconds)

CHAPTER 4

Indices and tables

- genindex
- modindex
- search