インテル® DAAL の Python* API (daal4py) 導入ガイド

この記事は、インテル® デベロッパー・ゾーンに公開されている「Getting Started with Python* API (daal4py) for Intel® Data Analytics Acceleration Library (Intel® DAAL)」の日本語参考訳です。

本ガイドでは、インテル® oneAPI のインテル® データ・アナリティクス・アクセラレーション・ライブラリー (インテル® DAAL) を使用した Python* API (daal4py) の導入に関する情報を提供します。

daal4py とは?

インテル® AI アナリティクス・ツールキット (英語) の Python* 向けインテル® ディストリビューション に含まれる daal4py は、マシンラーニングのアルゴリズムとフレームワークに優れたパフォーマンスを提供する Python* API です。データ・サイエンティスト向けに設計されており、強力なインテル® DAAL マシンラーニング・アルゴリズムを、柔軟かつカスタマイズ可能な方法で簡単に利用できるようにします。また、データの処理と分析には、バッチ、ストリーミング、および分散処理モードが用意されており、システムのニーズに応じて最適なオプションを選択できます。

daal4py の高速なフレームワークは、scikit-learn からマシンラーニング・アルゴリズムを加速する方法としてよく知られていますが、このガイドでは daal4py アルゴリズムを直接使用する方法を紹介します。

daal4py のインストール

daal4py パッケージをインストールする方法はいくつかあります。

1 つ目の方法は、多数の高速化された Python* パッケージとアプリケーションを含む、インテル® oneAPI の Python* 向けインテル® ディストリビューションの一部としてインストールします。ご使用のマシンに応じて、適切な手順 (英語) に従ってください。

2 つ目の方法は、次の Anaconda* コマンドを使用して、Python* 向けインテル® ディストリビューションの daal4py を直接インストールします。

```
conda install -c intelpython3_full python=3
```

3 つ目の方法は、インテル® AI アナリティクス・ツールキットの一部として daal4py をインストールします。この場合、daal4py は Python* 向けインテル® ディストリビューションの一部として、ハイパフォーマンスなマシンラーニングとディープラーニングのパッケージとともにインストールされます。

4 つ目の方法は、次の Anaconda* コマンドを使用して、daal4py パッケージを直接インストールします。

```
conda install -c intel daal4py
```

これで、daal4py と必要なパッケージがインストールされます。

daal4py の使用

バッチ処理

少量のデータの場合、バッチ処理モードを利用して入力データを一括入力することができます。バッチ処理は、daal4py のデフォルトの処理モードなので、実行するため daal4py コードを変更する必要はありません。

daal4py のバッチ処理のコード例

この例では、バッチ処理を使用して線形回帰モデルを作成し、それぞれの家の特徴からボストンの住宅の価格を予測するのに使用します。

最初に、必要なすべてのデータとパッケージをインポートします。必要に応じて、モデルと結果を格納するディレクトリーも作成します。

```
##### 共有メモリーシステム向けの daal4py 線形回帰の例 #####
import daal4py as d4p
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
import pickle
import os


# 必要なすべてのディレクトリーを作成
try:
    os.mkdir("./models")
    os.mkdir("./results")
except:
    pass
```

データセットをロードして、モデルで作業するのに必要な調整を行います。

```
# データをロード

data = load_boston()


# モデルで予測に使用する変数

X = data.data # house characteristics

y = data.target[np.newaxis].T # house price


# トレーニング用 (75%) とテスト用 (25%) にデータを分割

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25,
random_state =1693)
```

モデルをトレーニングして、予測に使用します。

```
# 予測用にモデルをトレーニング

train_result = d4p.linear_regression_training().compute(X_train, y_train)

# now predicting the target feature(s) using the trained model
y_pred = d4p.linear_regression_prediction().compute(X_test,
train_result.model).prediction
```

結果を出力します。

```
print(y_pred)
```

ストリーム処理

大量のデータの場合、バッチ処理とは異なり、すべての入力データを一括入力できないことがあります。例えば、データが複数のファイルに存在し、マージするにはコストがかかりすぎたり、データセットが大きすぎてメモリーに収まらない可能性があります。また、処理するデータが実際にストリームである場合もあります。daal4py のストリーミング処理モードは、このようなデータを素早く簡単に処理できるようにします。

ストリーミング処理は、特定のケースをサポートするだけでなく、I/O 操作と計算のインターリーブも可能にします。

daal4py のストリーミング処理のコード例

この例では、ストリーミング処理を使用して、線形回帰モデルを作成し、予測に使用します。この例を適切に実行するには、次のオープンソースのデータセットをダウンロードして使用してください。

ダウンロード

daal4py でストリーミング処理を使用する方法は、バッチ処理と同じですが、ファイルをチャンクとして読み取るようにコードを変更し、ストリーミング用の線形回帰トレーニング・オブジェクトを設定する必要があります。

```
# トレーニング用にチャンクとして読み取るファイル
infiles = ["./data/streaming_data/linear_regression_train_1.csv",
"./data/streaming_data/linear_regression_train_2.csv","./data/streaming_data/
linear_regression_train_3.csv","./data/streaming_data/linear_regression_train
_4.csv","./data/streaming_data/linear_regression_train_5.csv"]

# ストリーミング用の線形回帰オブジェクトを設定
train_algo = d4p.linear_regression_training(interceptFlag=True,
streaming=True)

# チャンクの読み取り
for file in infiles:
    indep_data = pd.read_csv(file).drop(["target"], axis=1) # house
characteristics
    dep_data   =pd.read_csv(file)["target"] # house price
    train_algo.compute(indep_data, dep_data)


# すべてのチャンクの読み取りが完了したので計算をファイナライズ
train_result = train_algo.finalize()
```

分散処理

daal4py は、SPMD (Single Program Multiple Data) 形式で動作し、(MPI のように) プログラムは複数のプロセスで実行されます。daal4py の SPMD モードの動作に MPI は必要ありません。必要なすべての通信と同期は daal4py で行われます。ただし、同じプログラムで daal4py と mpi4py を使用することは可能です。

daal4py をワークステーションのクラスター上で実行するために必要なコード変更はわずかです。

daal4py の分散処理のコード例

この例では、分散処理を使用して、線形回帰モデルを作成し、予測に使用します。この例を適切に実行するには、次のオープンソースのデータセットをダウンロードして使用してください。

ダウンロード

分散処理モードでコードを適切に実行するには、以下のコードを .py ファイルとしてダウンロードして、次のコマンドを実行します (コマンド中の「4」は、4 プロセスで実行することを意味します)。

```
mpirun -n 4 python ./linear_regression_spmd.py
```

分散処理モードでの daal4py の実行は、いくつかの点を除いて、バッチモードと似ています。

データをロードして、分散エンジンを初期化します。

```
# データセットを **ロード** して、モデルでの操作に必要な **調整** を行う
# 分散モードでは、各ファイルに固有の ID がある
# **分散エンジンを初期化**

d4p.daalinit() # initializes the distribution engine

# モデルで予測に使用する変数
# 各プロセスは個別のデータを取得
infile = "./data/distributed_data/linear_regression_train_" +
str(d4p.my_procid()+1) + ".csv"

# データの読み取り
indep_data = pd.read_csv(infile).drop(["target"], axis=1) # 家の特徴
dep_data   = pd.read_csv(infile)["target"] # 住宅の価格
```

分散モードでモデルをトレーニングします。

```
# **モデルをトレーニング** して、モデルの特徴を確認

# In[3]:


# 予測用にモデルをトレーニング
train_result =
d4p.linear_regression_training(distributed=True).compute(indep_data,
dep_data)
```

コードの最後で、分散エンジンを停止することを忘れないでください。

```
d4p.daalfini() # 分散エンジンを停止
```

daal4py の詳細については、daal4py のドキュメント (英語) を参照してください。

さらに高速なアプリケーション・パフォーマンスを実現するため、インテル® oneAPI をご利用ください。

製品とパフォーマンス情報

[1] インテル® コンパイラーでは、インテル® マイクロプロセッサーに限定されない最適化に関して、他社製マイクロプロセッサー用に同等の最適化を行えないことがあります。これには、インテル® ストリーミング SIMD 拡張命令 2、インテル® ストリーミング SIMD 拡張命令 3、インテル® ストリーミング SIMD 拡張命令 3 補足命令などの最適化が該当します。インテルは、他社製マイクロプロセッサーに関して、いかなる最適化の利用、機能、または効果も保証いたしません。本製品のマイクロプロセッサー依存の最適化は、インテル® マイクロプロセッサーでの使用を前提としています。インテル® マイクロアーキテクチャーに限定されない最適化のなかにも、インテル® マイクロプロセッサー用のものがあります。この注意事項で言及した命令セットの詳細については、該当する製品のユーザー・リファレンス・ガイドを参照してください。

注意事項の改訂 #20110804