インテル® DAAL を使用した Python* ナイーブベイズ・アルゴリズムのパフォーマンス向上

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「Using Intel® Data Analytics Acceleration Library to Improve the Performance of Naïve Bayes Algorithm in Python*」の日本語参考訳です。

はじめに

Netflix は視聴者が興味を持ちそうな動画をお勧めし、Amazon は潜在的な顧客に適切な商品を提案し、Microsoft Outlook* はスパムメールを分類します。

それぞれ、どのようにしているのでしょうか? Netflix は、視聴履歴から顧客に動画を勧めています。Amazon は、顧客の閲覧および購入履歴データを使用して、顧客が興味を持ちそうな製品を提案しています。Microsoft は、膨大な量のメールを分析して、メールがジャンクメールやスパムメールかを特定しています。

Netflix、Amazon、および Microsoft は、顧客のニーズに応えるため、過去のデータを分析しています。ソーシャルメディアやインターネットで膨大な量のデータ (ビデオ、オーディオ、テキスト) が利用できるようになったことで、人々がどのように考え、行動し、社会や環境と向き合っているか理解するため、これらのビッグデータを扱う人的介入を最小限に抑えた効率良い方法が必要になり、マシンラーニングが注目されています [1]。

この記事では、マシンラーニングとナイーブベイズ (NB) と呼ばれるマシンラーニングの手法/アルゴリズムにを説明します [2]。また、NB アルゴリズムのパフォーマンスを向上するインテル® 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) についても述べます [3]。

マシンラーニングとは?

マシンラーニング (ML) は、データセットに基づく解析モデルの作成に使用されるデータ解析手法です。解析モデルは、新しいデータが供給されると、明示的なプログラミングなしで学習することができます。ML はかなり前から存在していましたが、最近になって次の理由からその有用性が証明されました。

- ソーシャルメディアやインターネットで利用可能なデータの量と種類の増加
- コンピューター・システムの性能の向上
- データストレージの大容量化と低価格化

最も一般的なタイプの ML は、教師あり学習 [4] と教師なし学習 [5] です。

教師あり学習では、入力データのセットとラベル (既知の結果) のセットを使用してアルゴリズムをトレーニングします。アルゴリズムは、入力データの結果とラベルを比較するたびに学習し、マシンラーニング・モデルを調整します。分類は、教師あり学習と見なされます。

教師あり学習とは異なり、教師なし学習では、アルゴリズムが学習に使用できるラベルはありません。代わりに、入力データを調査して、自力でパターンを検出する必要があります。例えば、ある人物が世界のどの地域に属しているか分類するには、アルゴリズムは人口データを調査し、人種、宗教、言語などを特定する必要があります。

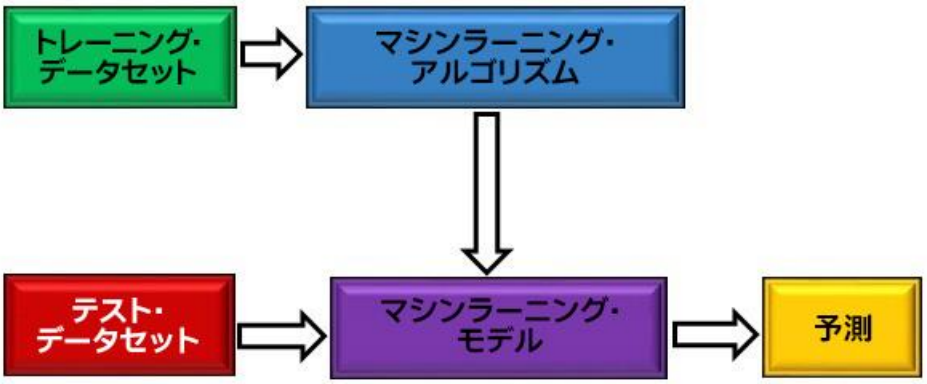

図 1: マシンラーニングの概略図

図 1 は、ML の仕組みの概略図です。最初に、トレーニング・データセットを使用して ML アルゴリズムをトレーニングして、ML モデルを作成します。ML モデルはテスト・データセットを処理して、最終的に結果を予測します。

次のセクションでは、教師あり学習アルゴリズムの 1 つである、ナイーブベイズ・アルゴリズムについて説明します。

ナイーブベイズ・アルゴリズム

ナイーブベイズ (NB) アルゴリズムは、ベイズの定理 [6] に基づく分類手法で、すべての特徴は互いに独立していると仮定します。

ベイズの定理は、次の式で表されます。

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

ここで、X と Y は特徴です。

- P(Y|X) は、X が与えられたときの Y の確率です。
- P(X|Y) は、Y が与えられたときの X の確率です。
- P(Y) は、Y の事前確率です。
- P(X) は、X の事前確率です。

この式 [2] は、次のように書き換えることができます。

$$P(Y|X) = P(x_1|Y) \times P(x_2|Y) \times \ldots \times P(x_n|Y) \times P(Y)$$

ここで、$X = (x_1, x_2, \ldots x_n)$ は n 個の特徴のベクトルを表します。

NB アルゴリズムは、メールのソート、ドキュメントの分類、スパムメールの検出など、一般的に使用されています。

図 2 は NB アルゴリズムの仕組みを示しています。

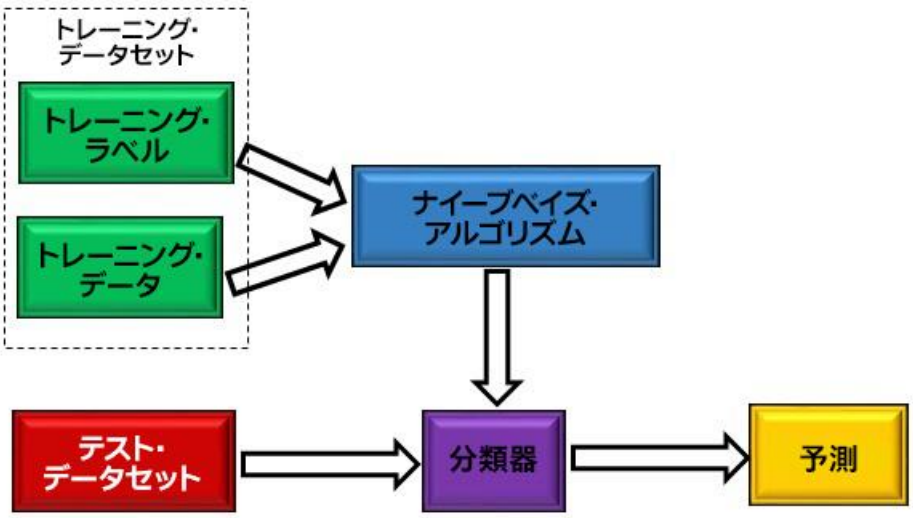

図 2: ナイーブベイズ・アルゴリズムを使用したマシンラーニングの図

図 2 から、トレーニング・データセットは、トレーニング・ラベルとトレーニング・データで構成されることが分かります。トレーニング・ラベルは、トレーニング・データの正しい出力です。分類器を作成には、この 2 つのセットが必要です。分類器を作成したら、評価のためテスト・データセットを分類器に供給します。

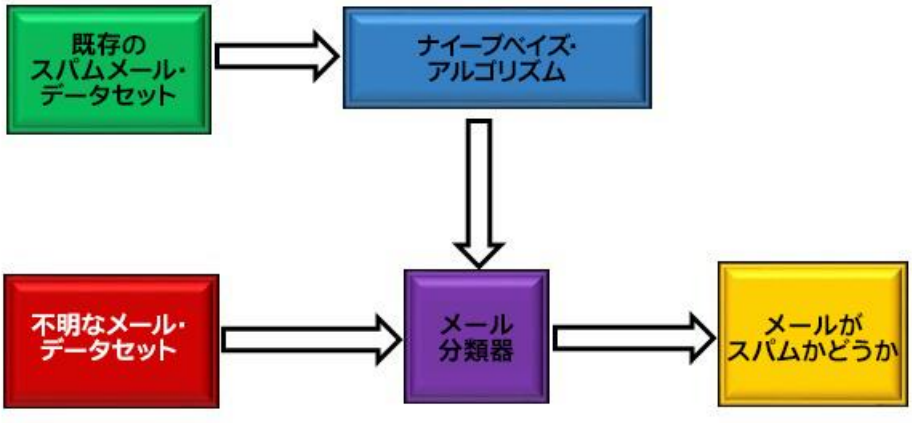

図 3: ナイーブベイズ・アルゴリズムを使用したスパムメールの検出

図 3 は、スパムメールを検出する NB アルゴリズムのフローです。メール分類器を作成するため、既知のスパムメールを NB アルゴリズムに供給します。分類器を作成したら、不明なメールをメールを分類器に供給してスパムかどうかチェックします。

NB の使用例

以下は、NB の使用例です。

- NB のスピードを利用したリアルタイム予測
- 花の分類などのマルチクラス/多項分類
- スパムメールの検出
- テキストの分類

NB のメリットとデメリット

以下は、NB のメリットとデメリットです。

メリット

- 素早くモデルをトレーニングできる
- マルチクラスの予測に優れている

デメリット

- トレーニング・データセットにラベルが含まれていない場合、予測できない
- 大きなデータセットの処理に適していない
- 特徴/イベントは常に完全に独立しているわけではない

大きなデータセットでは、モデルのトレーニングに長い時間がかかります。特定のモデルでは、数週間、あるいは数カ月を要します。トレーニングの最適化を支援するため、インテル® DAAL が開発されました。

インテル® DAAL

インテル® DAAL[7] は、データ解析とマシンラーニング向けに最適化された多くの基本ビルディング・ブロックからなるライブラリーです。これらの基本ビルディング・ブロックは、最新のインテル® プロセッサーの機能向けに高度に最適化されています。多項ナイーブベイズ分類器は、インテル® DAAL が提供する分類アルゴリズムの 1 つです。この記事では、PyDAAL[10] (インテル® DAAL の Python* API) を使用して基本的なナイーブベイズ分類器を作成します。

インテル® DAAL の NB アルゴリズムを使用する

このセクションでは、インテル® DAAL の Python*[9] 多項 NB アルゴリズム [8] を呼び出す方法を示します。

次のステップに従って、インテル® DAAL から NB アルゴリズムを呼び出します。

1. from コマンドと import コマンドを使用して、必要なパッケージをインポートします。
    a. 次のコマンドを実行して NumPy* をインポートします。

```
import numpy as np
```

b. 次のコマンドを実行して、インテル® DAAL の数値テーブルをインポートします。

```
from daal.data_management import HomogenNumericTable
```

c. 次のコマンドを実行して、NB アルゴリズムをインポートします。

```
from daal.algorithms.multinomial_naive_bayes import prediction,
training
from daal.algorithms import classifier
```

2. 入力データセットをトレーニング・データとテストデータに分割する関数を作成します。

   基本的に、入力データセット配列を 2 つの配列に分割します。例えば、100 行のデータセットを 80/20 (データの 80% をトレーニング用、20% をテスト用) に分割します。入力データセット配列の最初の 80 行がトレーニング・データになり、残りの 20 行がテストデータになります。

3. インテル® DAAL で読み取れるように、トレーニング・データセットとテスト・データセットを再構成します。

   次のコマンドを実行して、以下のようにデータを再構成します ("trainLabels" と "testLabels" は n × 1 テーブルとして扱います。ここで、n は対応するデータセットの行数です)。

```
trainInput = HomogenNumericTable(trainingData)
trainLabels =
HomogenNumericTable(trainGroundTruth.reshape(trainGroundTruth.shape[0],
1))
testInput = HomogenNumericTable(testingData)
testLabels =
HomogenNumericTable(testGroundTruth.reshape(testGroundTruth.shape[0],1)
)
```

   説明:

   `trainInput`: インテル® DAAL の数値テーブルに再構成されたトレーニング・データ。
   `trainLabels`: インテル® DAAL の数値テーブルに再構成されたトレーニング・ラベル。
   `testInput`: インテル® DAAL の数値テーブルに再構成されたテストデータ。
   `testLabels`: インテル® DAAL の数値テーブルに再構成されたテストラベル。

4. モデルをトレーニングする関数を作成します。
   a. 最初に、次のコマンドを実行して、モデルをトレーニングするアルゴリズム・オブジェクトを作成します。

```
algorithm = training.Batch_Float64DefaultDense(nClasses)
```

   b. 次のコマンドを実行して、トレーニング・データとラベルをアルゴリズムに渡します。

```
algorithm.input.set(classifier.training.data, trainInput)
algorithm.input.set(classifier.training.labels, trainLabels)
```

説明:

`algorithm`: 上記のステップで定義したアルゴリズム・オブジェクト。
`trainInput`: トレーニング・データ。
`trainLabels`: トレーニング・ラベル。

c. 次のコマンドを実行して、モデルをトレーニングします。

```
Model = algorithm.compute()
```

説明:

`algorithm`: 上記のステップで定義したアルゴリズム・オブジェクト。

5. モデルをテストする関数を作成します。
   a. 最初に、次のコマンドを実行して、モデルをテスト/予測するアルゴリズム・オブジェクトを作成します。

```
algorithm = prediction.Batch_Float64DefaultDense(nClasses)
```

   b. 次のコマンドを実行して、テストデータとトレーニング済みモデルをモデルに渡します。

```
algorithm.input.setTable(classifier.prediction.data, testInput)
algorithm.input.setModel(classifier.prediction.model,
model.get(classifier.training.model))
```

説明:

`algorithm`: 上記のステップで定義したアルゴリズム・オブジェクト。
`testInput`: テストデータ。
`model`: モデル・オブジェクトの名前。

6. 次のコマンドを実行して、モデルをテスト/予測します。

```
Prediction = algorithm.compute()
```

説明:

`algorithm`: 上記のステップで定義したアルゴリズム・オブジェクト。
`prediction`: テストデータの予測されたラベルを含む予測結果。

まとめ

インテル® DAAL アルゴリズムは、最新のインテル® プロセッサーの機能向けに高度に最適化されているため、インテル® DAAL アルゴリズムを使用することで、ML 関連のアプリケーションのパフォーマンスを容易に向上できます。また、新しいインテル® プロセッサーの機能を利用するため、コードを変更する必要がありません。インテル® DAAL の最新バージョンにリンクするだけで、最新のインテル® プロセッサーの機能を利用できます。

次の記事では、サポート・ベクトル・マシン (SVM) など、ほかのインテル® DAAL アルゴリズムを取り上げる予定です。

著者

Khang Nguyen
Zhang Zhang

関連情報

1. https://ja.wikipedia.org/wiki/%E6%A9%9F%E6%A2%B0%E5%AD%A6%E7%BF%92
2. https://ja.wikipedia.org/wiki/%E5%8D%98%E7%B4%94%E3%83%99%E3%82%A4%E3%82%BA%E5%88%86%E9%A1%9E%E5%99%A8
3. インテル® DAAL
4. https://ja.wikipedia.org/wiki/%E6%95%99%E5%B8%AB%E3%81%82%E3%82%8A%E5%AD%A6%E7%BF%92
5. https://ja.wikipedia.org/wiki/%E6%95%99%E5%B8%AB%E3%81%AA%E3%81%97%E5%AD%A6%E7%BF%92
6. https://ja.wikipedia.org/wiki/%E3%83%99%E3%82%A4%E3%82%BA%E3%81%AE%E5%AE%9A%E7%90%86
7. インテル® DAAL の概要 (英語)
8. https://en.wikipedia.org/wiki/Naive_Bayes_classifier#Multinomial_naive_Bayes (英語)
9. https://www.python.org/ (英語)
10. Linux* でインテル® DAAL の Python* バージョンをインストールする方法 (英語)
