インテル® DAAL を使用したサポート・ベクトル・マシン (SVM) の向上

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「Improving Support Vector Machine with Intel® Data Analytics Acceleration Library」の日本語参考訳です。

はじめに

インターネットが普及したことで、テキスト分類はテキストデータを扱い、整理する重要な手段になりました。テキスト分類は、ニュースを分類したり、ウェブ上で情報を検索するのに使用されます。また、ウェブ上で写真を検索したり、ライオンと馬を区別できるようにするには、写真を認識して分類するメカニズムが必要です。テキストや写真の分類には時間がかかります。このような分類は、マシンラーニング [1] に適しています。

この記事では、サポート・ベクトル・マシン [2] と呼ばれる分類マシンラーニング・アルゴリズムについて説明し、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)[3] を使用してこのアルゴリズムをインテル® Xeon® プロセッサー・ベースのシステム向けに最適化する方法を紹介します。

サポート・ベクトル・マシンとは?

サポート・ベクトル・マシン (SVM) は、教師ありマシンラーニング・アルゴリズムです。分類と回帰に使用できます。

SVM は、クラスが異なるオブジェクトの集合を分離する超平面 [4] を見つけることで、分類を実行します。この超平面は、ノイズを軽減し、結果の精度を高めるように、2 つのクラス間のマージンが最大になるように選択されます。マージン上にあるベクトルをサポートベクトルと呼びます。サポートベクトルは、マージン上のデータポイントです。

図 1 は、SVM がオブジェクトを分類する方法を示します。

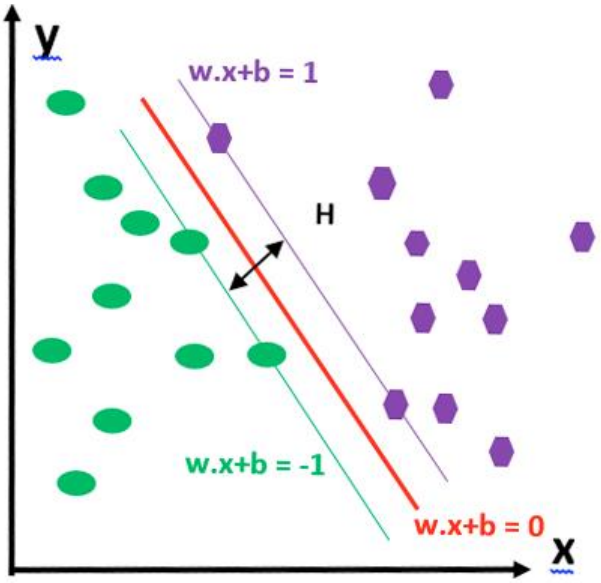

図 1: サポート・ベクトル・マシンを使用したオブジェクト分類

緑と紫の 2 つのクラスがあります。超平面はこの 2 つのクラスを分離します。超平面の左側にあるオブジェクトは、緑のクラスに属するものとして分類されます。同様に、超平面の右側にあるオブジェクトは、紫のクラスに属します。

前述のとおり、マージン H (2 つのマージン間の距離) を最大にして、ノイズを軽減し、予測の精度を向上する必要があります。

$$H = \frac{2}{|W|}$$

マージン H を最大にするには、|W| を最小にする必要があります。

また、2 つのマージン間にデータポイントがあってはなりません。そのためには、次の条件を満たす必要があります。

`xi •w+b ≥ +1` (`yi =+1` の場合)
`xi •w+b ≤ −1` (`yi =−1` の場合)

上記の条件は、つぎのように書き直すことができます。

`yi (xi •w) ≥ 1`

ここまでは、平面または 2 次元空間の線として超平面について述べました。しかし、実際には、必ずしもそうとは限りません。ほとんどの場合、超平面は直線ではなく曲線です (図 2)。

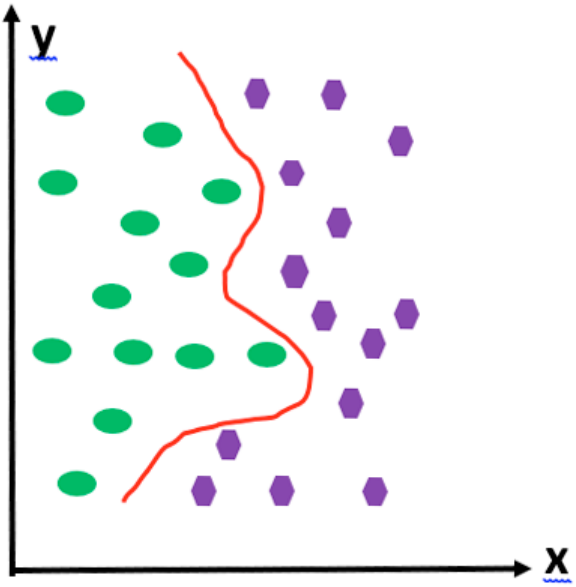

図 2: 曲線の超平面

便宜上、2 次元空間で作業していると仮定します。このケースでは、超平面は曲線です。曲線を直線に変換するため、全体を高次元に拡張できます。例えば、z という第 3 次元を追加して 3 次元空間にします。

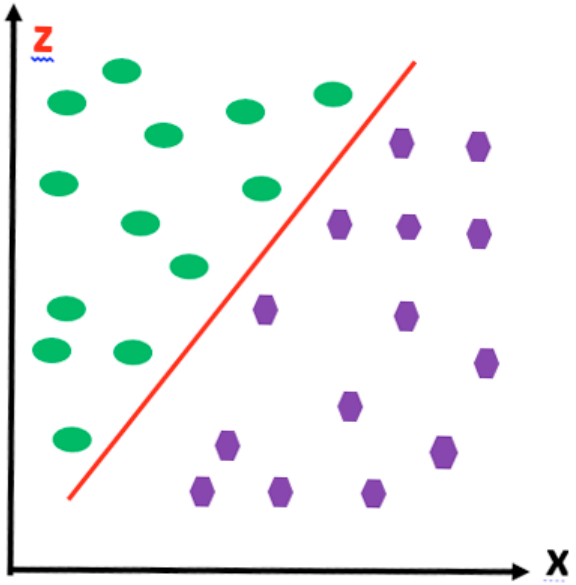

図 3: 第 3 次元 z の追加

データを高次元に拡張して直線や平面を作成する手法をカーネルトリック [5] と呼びます。

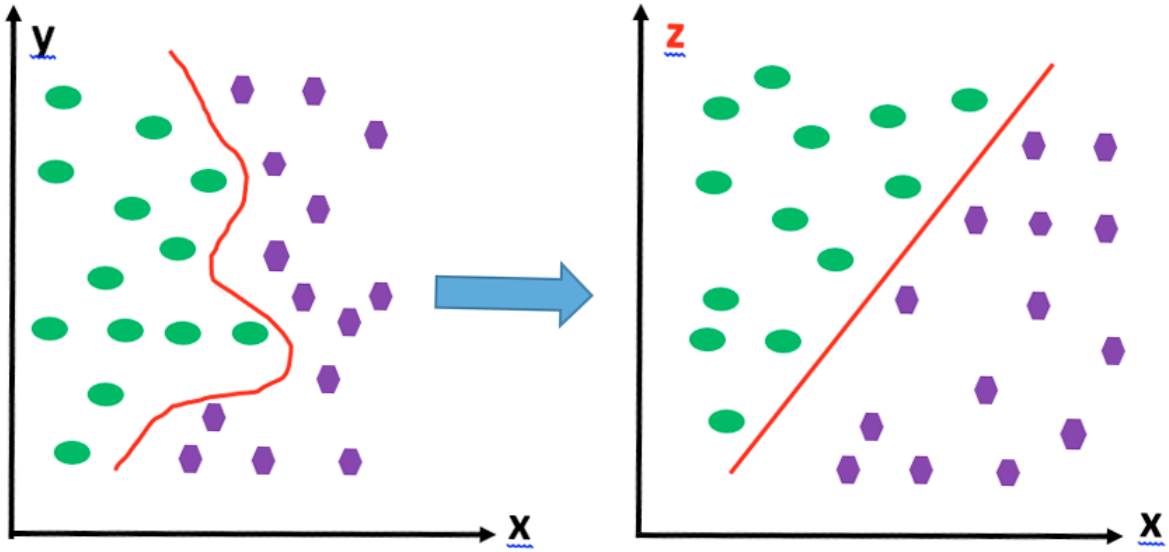

図 4: カーネルトリックを使用して高次元で直線または平面を作成する

SVM の使用例

以下は、SVM の使用例です。

- テキストとハイパーテキストの分類
- 画像の分類
- 手書き文字の認識

SVM のメリットとデメリット

以下は、SVM のメリットとデメリットです。

- **メリット**
  - 分離のマージンが明確な場合、最適に動作します。
  - 高次元空間で効果的です。
  - 決定関数 (サポートベクトル) でトレーニング・ポイントのサブセットを使用するため、メモリー効率に優れています。
- **デメリット**
  - 大きなデータセットでは、トレーニングに長い時間がかかります。
  - ターゲットクラス間に明確な分離がない場合、上手く動作しません。

インテル® DAAL

インテル® DAAL 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。SVM 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、インテル® DAAL の Python* API を使用して基本的な SVM 分類器を作成します。インテル® DAAL をインストールするには、ドキュメント [7] の手順に従ってください。

インテル® DAAL の SVM 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*[6] SVM アルゴリズム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 SVM 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。
    1. 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
        ```

    2. 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
        ```

    3. 次のコマンドを実行して、SVM アルゴリズムをインポートします。

        ```
        from daal.algorithms.svm import training, prediction
        from daal.algorithms import classifier, kernel_function
        import daal.algorithms.kernel_function.linear
        ```

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。

    次のコマンドを使用して、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。
    1. 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)
        ```

    2. 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。

    3. 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。
    1. 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)
        ```

    2. 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。

    3. 次のコマンド実行して、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

SVM は、強力な分類アルゴリズムです。分離のマージンが明確な場合、最適に動作します。インテル® DAAL の SVM アルゴリズムは最適化されています。インテル® DAAL を使用することで、アプリケーションを変更せずに、インテル® DAAL の最新バージョンにリンクするだけで、将来の世代のインテル® Xeon® プロセッサーの新機能を利用できます。

関連情報

1. https://ja.wikipedia.org/wiki/機械学習
2. https://ja.wikipedia.org/wiki/サポートベクターマシン
3. インテル® DAAL の概要 (英語)
4. https://ja.wikipedia.org/wiki/超平面
5. https://ja.wikipedia.org/wiki/カーネル法
6. https://www.python.org/ (英語)
7. 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)

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