参考資料

life.augmented

DT0139
設計のヒント

デシジョン·ツリーの生成

主要コンポーネント			
製品番号	**市場**	**ファミリ**	**機械学習コア（MLC）ノード数**
LSM6DSOX	民生用	iNEMO 慣性モジュール（IMU）	256 ノード
LSM6DSRX	民生用	iNEMO 慣性モジュール（IMU）	512 ノード
ISM330DHCX	産業用	iNEMO 慣性モジュール（IMU）	512 ノード
IIS2ICLX	産業用	2 軸加速度センサ	512 ノード

目的と利点

機械学習は、コンピュータ·アルゴリズムの研究分野の 1 つです。機械学習では、数学モデルを明示的に導き出すことなく、データを使用してアルゴリズムを構築することができます。広く使用されている機械学習の手法には、次の 4 つのタイプがあります。

- 教師あり学習
- 教師なし学習
- 半教師あり学習
- 強化学習

教師あり学習では、適切な実世界の情報でタグ付け（ラベル付け）された機械学習アルゴリズム用のトレーニング·データを使用します。その一例は、被験者の手首に装着されたデータ収集デバイスから収集できる加速度センサのデータです。この情報は、静止、歩く、走る、泳ぐ、自転車に乗るなど、さまざまな活動に対してラベル付けすることができます。教師なし機械学習アルゴリズムは、ラベル付きデータを必要とせず、データ内のパターンを特定することでモデルを作成できます。
半教師あり学習では、ラベル付きデータとラベルなしデータを組み合わせて使用し、トレーニングして精度を向上させます。強化学習アルゴリズムは、実世界との相互作用を通じてモデルを導き出し、改善します。機械学習アルゴリズムの詳細については、Wikipedia の機械学習に関するページが参考になります。

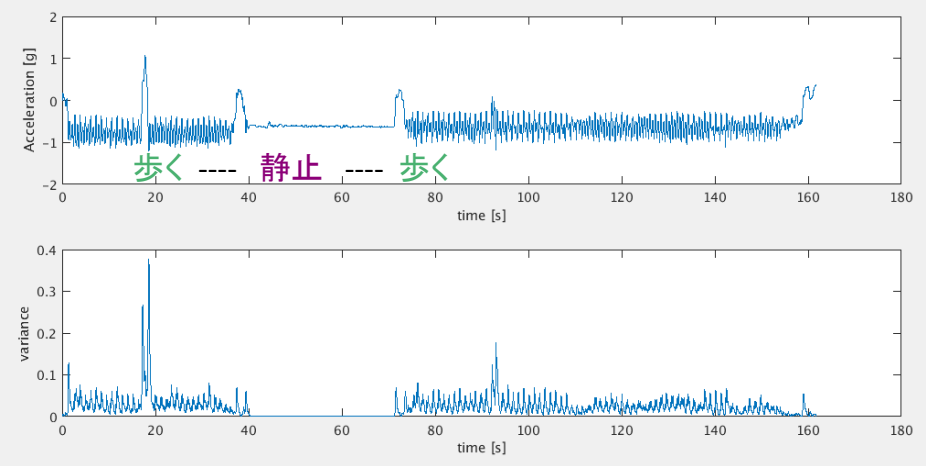
参考資料

説明

本文書では、1 つの教師あり学習アルゴリズムについて説明します。ST の最新世代の MEMS センサは、デシジョン·ツリー分類器に基づく機械学習コア（MLC）を内蔵しています。これらの製品は、製品名末尾の「X」によって容易に識別できます（例えば、LSM6DSO**X**）。この MLC は、超低消費電力のセンサ内でプログラムされたデシジョン·ツリーを実行することができます。これらのデバイス内の機械学習コアの詳細については、関連するアプリケーション·ノート（LSM6DSOX については AN5259、LSM6DSRX については AN5393、ISM330DHCX については AN5392、IIS2ICLX については AN5536）をご覧ください。

機械学習コアを備えたセンサは、センサ·データを処理し、RAW データおよびフィルタリングされたセンサ·データに対して平均、分散、帯域内のエネルギー、ピーク·ツー·ピーク値、ゼロクロス（正および負）、ピーク検出（正および負）など、データ·ウィンドウ上の 12 種類の特徴量を計算することができます。選択した特徴量に識別パターンが見られる場合は、デシジョン·ツリーを使用して分類を実行できます。図 1 に示す加速度データの例を考えてみましょう。

図1. 加速度データとその分散



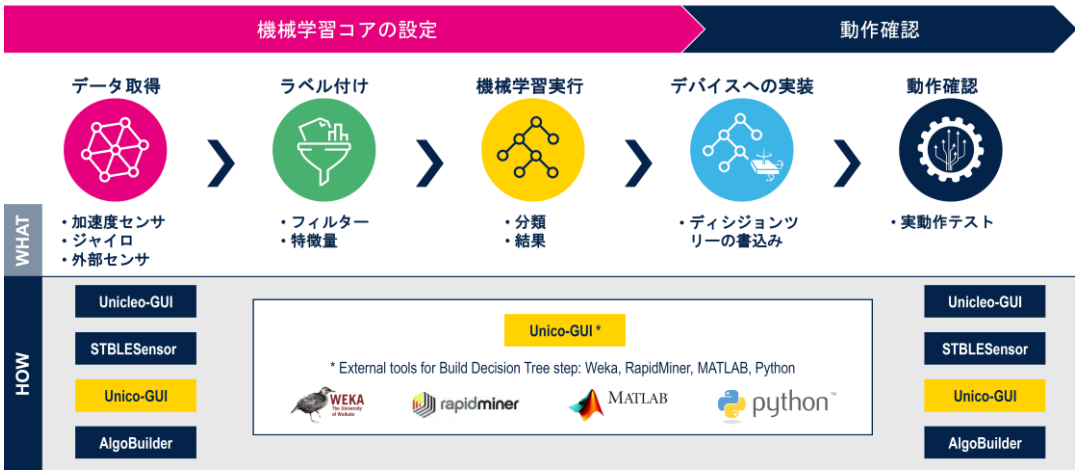
特定のウィンドウの長さにわたる加速度の大きさの分散（この場合の特徴量）をプロットしたものを上の図 1 に示します。2 つ目のサブプロットから、分散が「歩く」と「静止」の 2 つのクラス間で明確に異なる値を示していることが明らかです。この場合、これら 2 つの異なるタイプの動きを区別する特徴量の 1 つとして、分散を使用したデシジョン·ツリーを設計できます。

機械学習の分類アルゴリズムを開発する際の通常のフローを図 2 に示します。

図2. 機械学習アルゴリズムを開発する際の基本的なプロセス

データ収集とデータのラベル付け（教師あり学習の場合）は、最初の 2 つのステップです。次に、トレーニング・データ（収集データ全体のサブセット）を使用してモデルがトレーニングされます。モデルは最終的に、組込みハードウェアを使用してリアルタイムの検証テストを実行するために MLC コアを備えたデバイスに転送され、実装されます。

図3. 機械学習コアを実装するプロセスの詳細



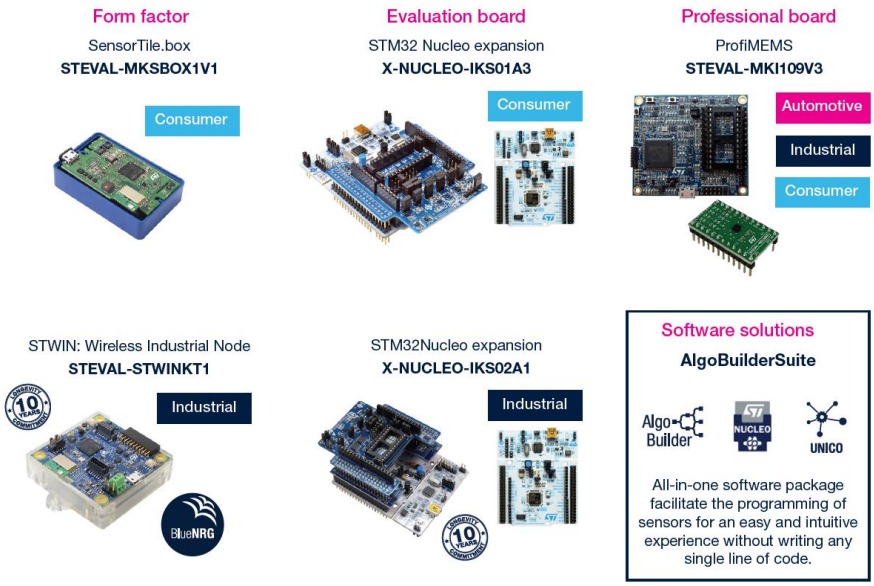
MLC の処理ステップと利用可能なツールのいくつかを図 3 に示します。この場合、最初の 2 つのステップは、例えば ST Unico-GUI を使用してデータを収集し、データにラベルを付け、各クラスを特定のログ・ファイルに保存することで対処できます。次に、WEKA、RapidMiner、MATLAB、Python（sklearn など）、Unico-GUI の組込みデシジョン・ツリー生成機能など、利用可能なツールのいずれかを使用して、デシジョン・ツリー・モデルを生成することができます。

本文書の最初のセクションでは、データの収集と前処理について説明します。 セクション 2 では、モデルを構築するために、トレーニング・データセットをさらに管理する方法と、システムをセットアップする方法（特徴量選択、ウィンドウ長の定義、センサの ODR など）について説明します。セクション 3 では、デシジョン・ツリーの生成の詳細について説明します。セクション 4 では、モデルを評価する方法と分類性能を最適化するための手法について説明します。

1. データ収集と前処理

機械学習（ML）の分類に関するあらゆる問題の最初のステップは、データを収集することです。センサ·データの収集とラベル付けは、PC 上の Unico-GUI、ST の BLE センサ·アプリ、AlgoBuilder、FP-SNS-DATALOG1 などのさまざまなアプリケーション、および ProfiMEMS ボード、SensorTile.Box、Nucleo ボード、STWIN といった各種のハードウェア·デバイス·オプションを使用して実行できます。

図4. 機械学習コアを利用するための開発キットと GUI



適切なデータ収集活動の一般的なガイドラインは、以下のポイントに基づきます。

- **クラス定義**: 分類のポイントは、認識可能なクラスとして定義する必要があります。さらに、入力が定義されたクラスのいずれにも属していない場合にどうなるかを明確にする必要があります。この場合、クラスのリストに追加として「その他/アイドル」クラスを含めることを検討します。
- **センサ**: 分類の問題には、加速度センサの信号を使用することで解決できるものもあれば、ジャイロセンサ信号を使用するもの、両方のセンサ入力を必要とするものもあります。6 軸慣性モジュール製品の場合は、両方のセンサからのロギングを有効にして、どの構成が最高の精度を実現できるかを調査することが推奨されます。3 軸地磁気センサには、6 軸 MLC ファミリのセンサのセンサ·ハブ機能によってアクセスすることもできます。
- **センサの設定**: 機械学習コアの設定ツール（Unico-GUI）では、センサの出力レート（ODR）やフルスケール（FS）など、MLC のすべてのパラメータを設定できます。
  - 必要な情報を取得するために十分な ODR と帯域幅でセンサ·データが収集されるようにします。帯域幅は、システムによって取得される最大周波数信号に相当し、通常、ODR の半分（BW ≒ ODR/2）です。正確な ODR の選択を事前に実現できない場合は、高い ODR（104Hz など）の最大帯域幅を設定した後、デシメーション/リサンプル方式を使用することが推奨されます。定義された構成をプロセス全体で維持する必要があること、および ML アルゴリズムがフィールド·データを同じ方法で処理することも考慮します。

- フルスケール（FS）は、アプリケーションの要件に基づいて定義する必要があります。例えば、センサが車両や手首用のアプリケーションに使用される場合、ジャイロの FS は大きく異なる可能性があります。

センサの設定を変更することでソリューションを最適化できます。

- **データの整合性とバランシング**：アプリケーションの要件に基づき、実際のユース･ケースに相当する十分なデータを収集します。アプリケーションが手首用の機器に基づいている場合、スマートフォンで記録されたデータは含めません。**各クラスに対して同様な量のデータを収集することも重要です。機械学習のトレーニング･アルゴリズムは、最大限の総合精度を達成しようとします。その結果、データ量が少ないクラスは重みが小さくなり、モデルはデータ量が多いクラスに偏る可能性があります。**例えば、クラス「A」のデータが 90%で、クラス「B」のデータが 10%の場合、モデルはすべての出力をクラス「A」に割り当て、精度は 90%になります。
- **データ･ロギングの形式**：Unico-GUI 以外のツールを使用してデータ･ロギングを行う場合は、Unico-GUI で定義されているものと同じ規則（詳細については、アプリケーション･ノート AN5259 を参照）に従って、データをテキスト･ファイル（.txt）に記録する必要があります。

データ収集の最初のフェーズの後、分類の問題を設定するために一連の操作が必要です。外れ値やラベル付けの不備を回避するために、データの検証とクリーニングを行う必要があります。データの可視化は、データの検証、特徴量の選択、ラベル付けなどのための優れた手法です。 次のセクションでは、これらの操作について説明します。

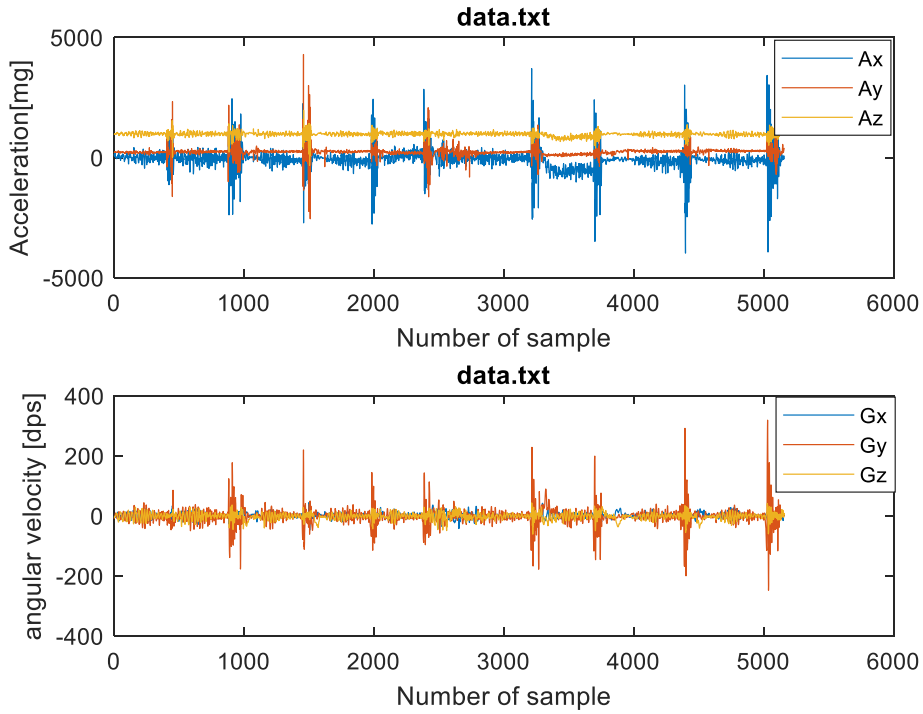
1.1 データの可視化

これは、ML モデルの開発において最も重要なステップの 1 つです。データの可視化は、データの前処理と分析で使用することができます。また、各種クラスにわたるさまざまな特徴とバリエーション間の関係を理解する際にも使用できます。このプロセスは、以下のように時間ドメインと周波数ドメインに分けることができます。

時間ドメインにおける信号解析

収集されたデータを可視化するために利用できるいくつかのオプションがあります。加速度センサのデータ[mg]とジャイロセンサのデータ[dps]を収集する場合、次の例に示すように、MATLAB/Python/Excel/R を使用してプロットを生成できます。

図5. 時間ドメインにおけるデータの可視化



時間ドメインにおける信号解析では、次のことが可能です。

- 低品質なデータや外れ値を視覚的に検査する。
- 各種クラス間にパターンが存在するかどうかを確認する。
- 有意な特徴量（平均、ピーク·ツー·ピーク、ピークなど）を選択する。

周波数ドメインにおけるデータ解析

周波数ドメインの解析は、時間ドメインの解析を補完する上で役立つ可能性があります。同じツール（MATLAB/Python/Excel/R）を使用して、FFT またはスペクトログラムのプロットを計算し生成することができます。このようなプロットから、信号を処理して特徴量を計算する前に、信号に対するフィルタの適用について有益な知見が得られることがあります。また、これらのプロットは、特定の記録されたログが正しく参照されているかどうかを評価する上で役立ちます。例えば、「歩く」のユース·ケースのデータ·ログについて考えてみましょう。FFT プロットを使用すると、処理された信号に対応する支配的な周波数を確認できます。この周波数が高すぎる場合（例えば、2.5Hz）、ログは「歩く」ではなく、「走る」として参照される必要があると見なすことができます。

図 6 は、分類の問題で考慮される人間の各種活動（歩く、速く歩く、ジョギングする、自転車に乗る）に対する周波数スペクトルを示しています。

図6. 各種活動に対する周波数ドメインの解析

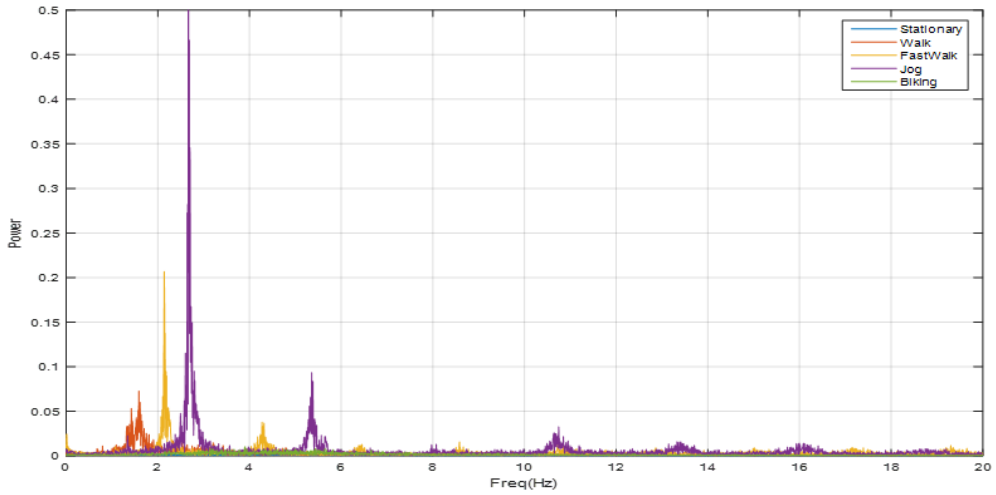


図 6 に示すプロットから、以下のような情報が導き出されます。各活動は、支配的な周波数成分がそれぞれ異なるため、適切なフィルタを適用して、各クラスの特徴的な信号を抽出することができます。例えば、1.8Hz でローパス·フィルタを適用すると、「歩く」クラスを、より高い周波数でより大きなエネルギーを特徴とする「速く歩く」クラスや「ジョギングする」クラスから分離できます。

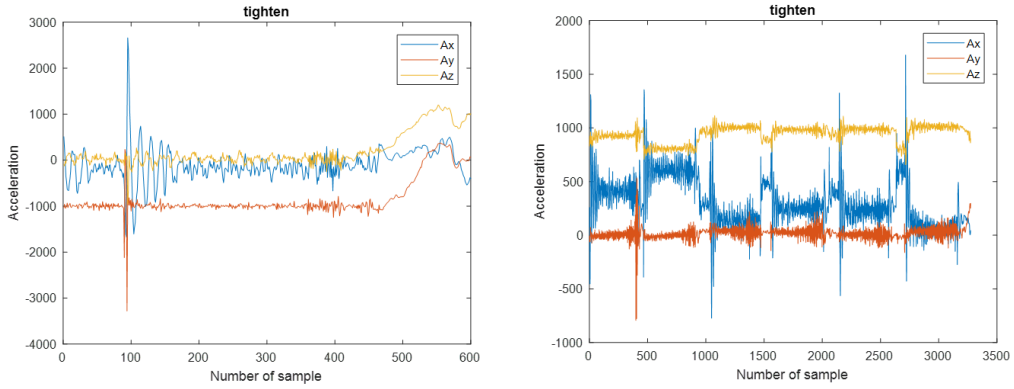
1.2 データのクリーニング

センサ·データのラベル付けは、機械学習アルゴリズムの開発における出発点です。最もよい方法は、ML のトレーニングを開始する前に、収集されたデータを検証し、健全性チェックを実施することです。データのクリーニングは、破損したデータ、不正な形式のデータ、不完全なデータなどを修正または削除するプロセスです。ユーザは、例えば平均値、分散値、最小値、最大値を簡単に可視化して確認することができます。計算された特徴量が同じラベル付きクラスの他のログと大幅に異なる（が、それらは当該アプリケーションで類似するものと想定されている）場合、そのラベル付きデータはクリーンではない可能性があります。

大半のプロジェクトでは、データを記録する際に正しい状態にあることを確認して収集します。通常、ログの開始および停止は、1 つのクラスのみに対応した状態で可能です。しかし、さまざまな理由により、1 つのログ·ファイルに複数の異なるクラスが含まれる場合があります。そのような場合、そのログ·ファイルが ML のトレーニングに使用できると見なされるためには、データを分割して、1 つのファイルに 1 つのクラスのみが含まれるようにする必要があります。

ユーザがよく犯す一般的な間違いの 1 つは、同一ファイルに複数のクラスが含まれたデータのログをそのまま使用してしまうことです。教師あり学習では、対応するクラスに属さないデータがあると、トレーニング済みモデルの性能に影響を与えるため、データのクリーニングは非常に重要です。例えば、下記のプロット（**Error! Reference source not found.**7）は、「締め付け」クラスについてデータを収集する際に、電動ドリルに固定された加速度センサからのデータを示しています。ここでは、目的は各クラス（アイドル、締め付け、緩め、ドリル）のラベル付きデータを収集することです。

図7. 電動ドリルから収集されたデータの例



左側のプロットは、電動ドリルの動きが停止した時にロギング・セッションは停止しないことを示しています。ドリルの動作がないプロットの最後には、余分なデータがあります。右側のプロットは、ログに複数のネジの締め付けに対するデータが含まれており、複数の異なるセッションが同一のファイルに記録されていることを示します。電動ドリルの向きは変化し、Ax、Az は 500～1000 サンプル間隔で変化します。これらの問題はすべて、生成されたデシジョン・ツリーの性能に影響を与える可能性があります。いくつか提案するとすれば、次のとおりです。

- ラベル付きデータの単位は、各ファイルで一貫している必要があります。本文書では加速度に[mg]、角速度に[dps]を使用しています。
- データ収集の各セッションは、適切な期間だけ切り抜く必要があります。
- Python、MATLAB、R などのツールを使用して、クラスに基づいてデータを分離します。
- どのクラスにも属さないデータは削除するか、または「その他/アイドル」クラスに使用します。
- 必要に応じてスケーリングを修正します。

2. ウィンドウ・サイズと特徴量の選択

収集されたデータの前処理が完了した後、次のステップはウィンドウの長さを決定し、特徴量選択のプロセスを開始することです。

2.1 ウィンドウ・サイズ

ウィンドウの長さは、アプリケーションに大きく依存します。ウィンドウは、デシジョン・ツリーで異なるクラスを分離するためのすべての情報を取得できるように設定する必要があります。ウィンドウの長さが短いと、ウィンドウの長さごとに予測が行われるため、応答が速くなる可能性があります。しかし、ウィンドウの長さが短すぎて十分な情報を取得できなければ、精度が低下する可能性もあります。ウィンドウのサイズは、データ分析と可視化の後にユーザが慎重に選択する必要があります。動きが特定の周波数で繰り返される場合、最適なウィンドウ・サイズとしては、動きのパターン全体を捕捉するために十分な長さである必要があります。極めて緩慢に変化する信号も考慮してください。例えば、周期性が 0.9Hz のクラス A と周期性が 1.9Hz のクラス B を識別する場合は、クラス A に基づいてウィンドウ・サイズを決定し、ウィンドウ・サイズを（1/0.9Hz）≈ 1.1 秒以上に設定する必要があります。

2.2 特徴量の選択

2.2.1 観察

動きを観察することによって、妥当性のある有意な特徴量を選択できます。センサがヘッドセットに取り付けられ、y 軸が地面（重力方向）を指し、x 軸がユーザの進行方向と一致している時に、「うなずく」などの頭のジェスチャを検出する例を考えてみましょう。この場合、x 軸と y 軸の加速度とその分散、平均値、最小値、最大値を選択することが考えられます。ユーザがうなずくと、x 軸と y 軸についてこれらの特徴量が変化するからです。

方向に依存しない特徴量を使用する別の例を取り上げましょう。電動ドリルにセンサを取り付ける場合、電動ドリルの使用方向によってこの機械の用途を制限することは望ましくありません。この場合、x、y、z、平均値、最小値、最大値など、方向に依存する特徴量は適用しません。分散、ゼロクロス、ピーク検出、エネルギーの大きさ（x、y、z 軸の大きさ）を選択することが考えられます。これらの特徴量は、電動ドリルの向きに依存せず、振動の検出に最適です。

2.2.2 モデルのトレーニングの活用

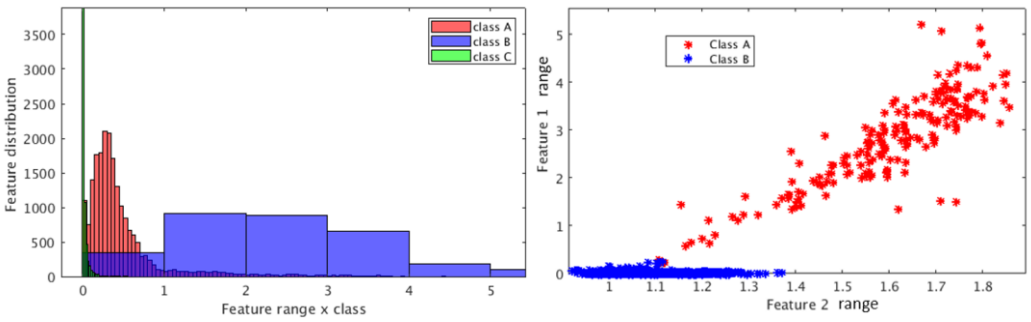
特徴量の選択について可能性がある最初の候補群を導き出すための 1 つの手法は、利用可能なすべての特徴量を選択し、選択した分類器に基づいてモデルをトレーニングすることです。この場合、利用可能な特徴量は MLC で事前に定義されており、分類器はトレーニングする必要があるデシジョン·ツリーです。モデルのトレーニング中に、デシジョン·ツリーの学習アルゴリズムは、クラスをより適切に識別できる特徴量を選択します。また、問題を解決するために直感的な特徴量を選択することが推奨されます。これらの特徴量はその後、機械学習モデルの開発における今後のステップで使用するための唯一の特徴量として選択することができます。

この手法にはいくつかの欠点があります。例えば、あまりにも多くの特徴量や特定のクラスの検出に実際には関係のない特徴量が使用され、過学習したモデルが得られる可能性があることなどです。そのため、特徴量の選択においては他の手法にも依存し、場合によってはそもそもすべての特徴量を使用しないようにすることが常に推奨されます。より優れた手法は、まず適切と考えられる一連の特徴量を特定し、次にモデルをトレーニングして、どの特徴量が使用されるかを確認することです。

2.2.3 特徴量の可視化

可視化は、モデルの性能を最適化するための特徴量の選択や削除に役立つ場合があります。1 つの特徴量からの情報の寄与は、さまざまなクラスがどれだけ分離されているかを測定することで評価できます。この分析は、**Error! Reference source not found.**に示すような 1D/2D 表現によって視覚的に実行することができます。

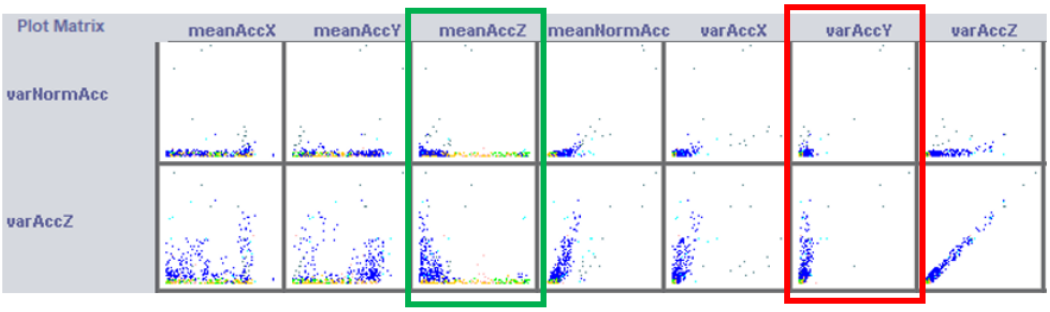
図8. 3 つの異なるクラスの単一の特徴量に関する計算値のヒストグラム。これらは適度に分離されています。つまり、選択された特徴量は、さまざまなクラスを識別する上で有用です。



可能な場合、上の図（右側）のように、2 つの特徴量のデカルト図をプロットすると役立つ場合があります。ここには、2 つのクラスの分類問題と 2 つの異なる特徴量の選択に関連する 2D プロットが示されています。この場合、厳密な分離は明らかであり、したがって選択された 2 つの特徴量の組合せによる情報が表示されています。この種のプロットは、さまざまな特徴量とクラス間の関係を評価する際にも有用です。左側の図は、各クラスの 1 つの特徴量のヒストグラムです。各クラスの特徴値の分布は非常に異なり、単純な閾値を使用してもクラスを分離できることがわかります。

同じ手法は WEKA などの ML ツールでサポートされています。以下に示すプロットは、さまざまな特徴量とクラス間の関係を示しています。特に、緑色の線で囲まれた特徴量の組合せは分離可能性を示していますが、赤色の線で囲まれた特徴量の組合せは分離可能ではないことがわかります。

図9. WEKA における特徴量の可視化

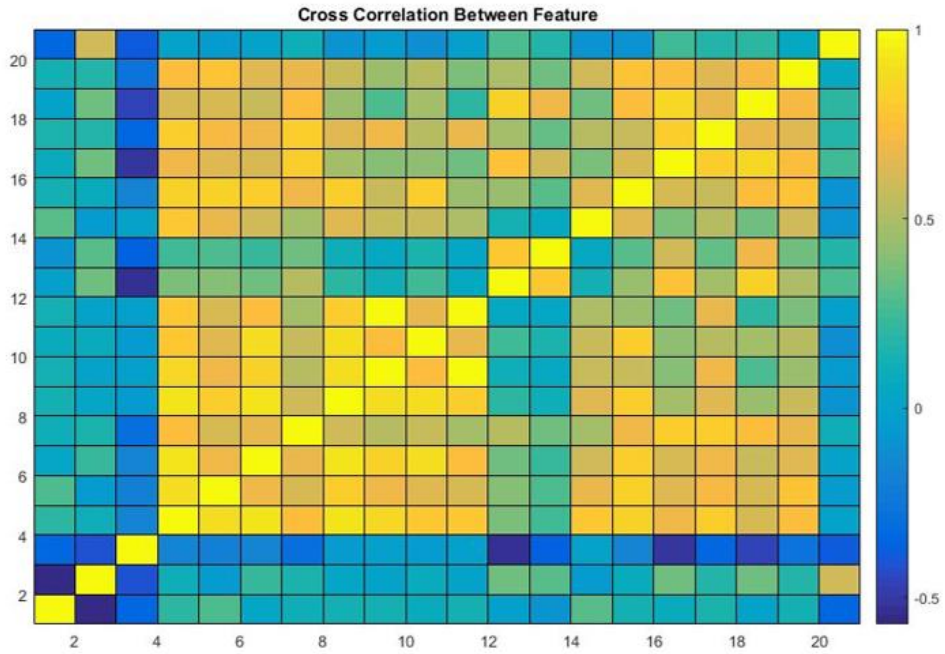


特徴量を最終的な計算に含める前に、次の点を確認します。

- ある特徴量が別の特徴量と線形関係にあるか、強く相関している場合、情報はすでに別の特徴量によって捕捉されているため、その特徴量は削除できます。可視化が難しい場合は、図 10 の例に示すように、共分散行列を計算して、最も相関がある特徴量を特定できます。このプロットでは、強く相関しているために削除できる非対角要素が多数あることがわかります。

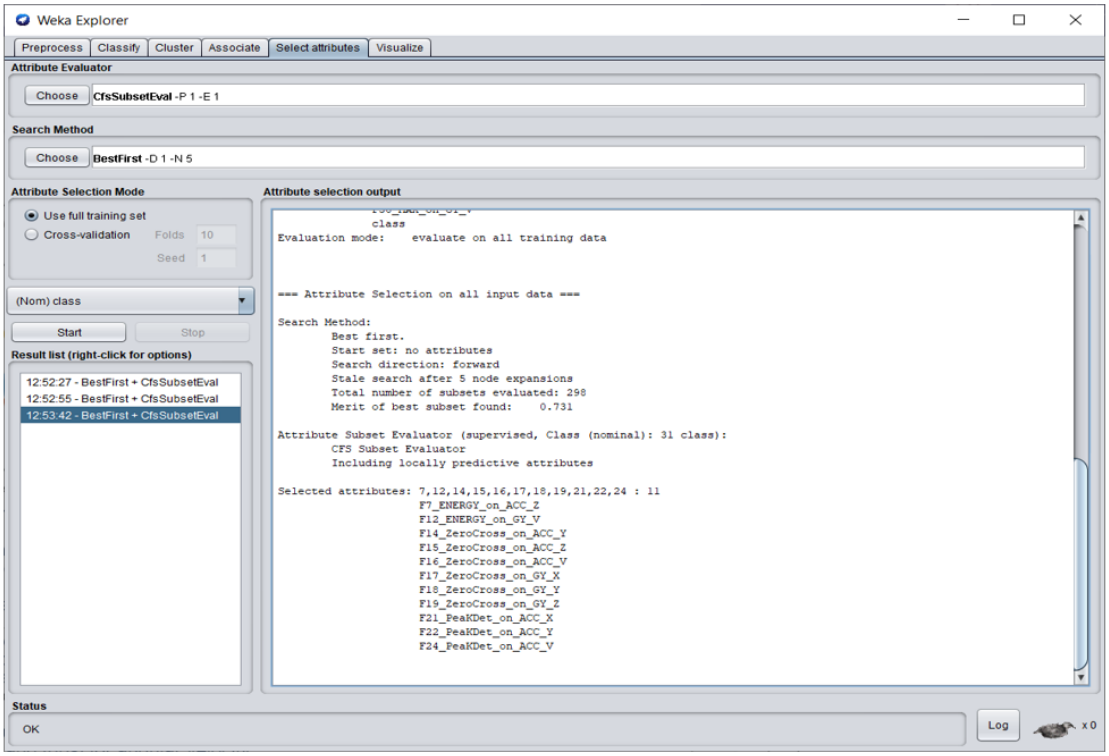
- 変動性：特徴量がさまざまなクラス間であまり変化を示さない場合、その特徴量はクラスを分離する上で役立たない可能性があります。

図10. 特徴間の相互相関



最後に、WEKA は、有用な特徴量の最善のサブセットを識別するためのいくつかの組込みツールを提供します。図 11 には、Weka で特徴量のランク付けを取得するプロセスの例を示しています。ここに示した例では、ARFF ファイルに 30 の特徴量があり、特徴量のランク付けロジックを使用することにより、Weka は現在のデータセットに対して 11 の重要な特徴量のみを選択しています。

図11.　Weka で利用可能な特徴量選択の手法

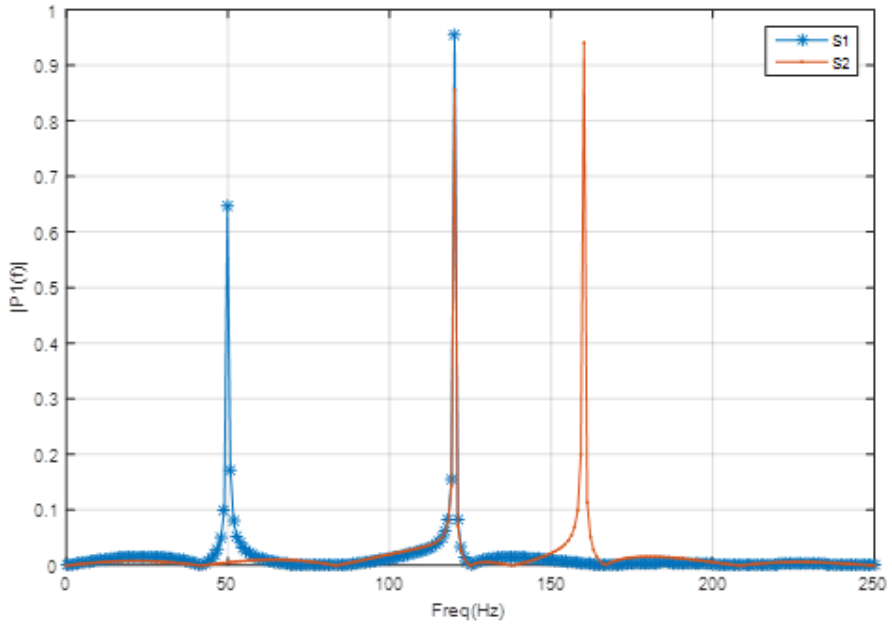


2.3　フィルタ選択

平均値、分散、エネルギーなどの基本的な特徴量には直接的な情報が含まれていますが、支配的な周波数が異なるクラスを分離できない場合があります。信号にフィルタを適用すると、信号から特定の情報を分離でき、クラスを適切に分離できる場合があります。RAW データに専用のフィルタを適用することにより、特定の周波数領域を選択できます。人間の活動を検出する場合、適切なフィルタを使用することで、ユーザが歩いているか（通常は 1～2Hz）、ジョギングしているか（2.5～4Hz）を正確に識別することができます。フィルタの選択は、フーリエ解析を行い、信号のパワー·スペクトル密度を観測することによって行う必要があります。フーリエ解析は、各クラスでどの周波数領域が支配的であるかについての知見を提供します。

このプロセスでは、適切なフィルタ（ローパス、ハイパス、バンドパス）が選択され、カットオフ周波数が定義されます。フィルタと対象の周波数の選択は、クラス間の重複しない領域で大きなエネルギーを見つけることによって行われます。次の図 12 は、クラス S1 と S2 の重複領域と対象の周波数領域を示しています。

図12.　2 つの異なるクラスに関する周波数解析



総エネルギー（すべての周波数（0～250Hz）にわたる総パワー）を計算する場合、特定の周波数（50Hz など）でのエネルギーに関する情報は提供されません。したがって、さまざまなクラスについて総エネルギーが類似している一方、異なる周波数ではパワーが異なる場合は、2 つのクラスを分離するだけでは不十分な場合があります。ユーザは、エネルギーを計算する前に、適切なフィルタを適用して特定の周波数領域から信号を除去する必要があります。フィルタを適用すると、信号は（ローパス･フィルタの場合）カットオフ周波数よりも低い閾値を通過できます。図 12 の例を取り上げ、60Hz でローパス･フィルタを適用した場合、信号 S2 はフィルタで完全に除去され、55Hz 未満の S1 の一部が通過します。ここで、フィルタ適用後に両方の信号のエネルギーを計算すると、信号 S2 はフィルタで完全に除去されるため、S1 のエネルギーは S2 に比べて高くなります。上記の場合に推奨されるフィルタは、50Hz および 150Hz のバンドパス･フィルタ、または 55～100Hz および 125～150Hz のローパス･フィルタです。

2.4　トレーニング・データとテスト・データの分割

ウィンドウの長さを決定したら、次のステップは、データをトレーニング･セットとテスト/検証セットに分割する方法を決定することです。すべての機械学習モデルについて、テスト、トレーニング、および検証（必要な場合）にデータを分割することが重要です。モデルを選択し、最終モデルの品質（学習不足、過学習）を確認する上では、データの分割が必要です。トレーニング･データはモデルのトレーニングに使用され、テスト･データはトレーニングされたモデルの品質/精度の評価に使用され、検証データは適切なモデルを選択するために複数のモデルがある場合に使用されます。テストおよびトレーニング･データの精度は、モデルが学習不足、過学習、または適性学習であるかどうかを判断する上で役立ちます。例えば、トレーニング・データに対する精度は高く（80%超）、テスト･データについては低い（70%未満）場合、モデルは過学習である可能性が高く、いくらかの剪定やより多様性のあるデータの追加を必要とする場合があります。一方、トレーニング・データとテスト･データに対する精度がともに低い場合（60%未満）、モデルは学習不足である可能性が高

く、特徴量の追加や剪定の削減が必要になる場合があります。詳細については、セクション 4 を参照してください。

通常、1 つのモデルを評価する場合、トレーニング･サンプルとテスト･サンプルの比率は約 80:20 または 70:30 です。 複数のモデル（異なるトレーニング･アルゴリズム J48、CART、または異なる特徴量セットに基づく）の評価を行う場合は、データを 60:20:20 の比率でトレーニング、検証、およびテストに分割する必要があります。検証データは、最適なモデルを選択するためにのみ使用する必要があり、テスト･データは、選択したモデルの最終的な精度を評価するために使用する必要があります。データを分割する比率の選択は問題に依存し、アプリケーションの要件に基づいて変更できます。

次に示す Python のコードは、データ全体をトレーニング･データとテスト･データに分割するものです。

```
from sklearn.model_selection import train_test_split
# split train : test = 70 : 30
X_train, X_test, y_train, y_test =
  train_test_split(wholeData, label, test_size=0.3)

# split train : test : validate = 70: 15 : 15
X_test, X_validate, y_test, y_validate =
    train_test_split(X_test, y_test, test_size=0.5)
```

上記の Python のコードは、まずデータセット全体を 70:30 の比率でトレーニング用とテスト用に分割します。次に、テスト･データセットを半分に分割して片方を検証データセットとします。トレーニング、テスト、および検証の最終的な比率は、70%、15%、および 15%とすることができます。

検証データセットを分割する必要があるかどうかについては、これを判定するための簡単なルールがあります。モデルが 1 つの場合は、データをトレーニングとテスト（30%-70%、または 20%-80%）に分割し、トレーニング･データでトレーニングされたモデルをテスト･データセットで評価します。複数のモデルがある場合は、データをテスト、検証、およびトレーニング･データセット（15%-15%-70%、または 20%-20%-60%）に分割します。トレーニン･データグと検証データで相互検証を実行することができます。次に、検証データを使用してモデルを選択し、最終的なモデルをテスト･データセットで評価します。

3 デシジョン·ツリー

デシジョン·ツリーは、ノードとリーフ(子を持たないノード)で構成されるバイナリ·ツリー構造です。ノードには 2 本の分岐があります。リーフは、分類出力または決定を生成します。各ノードでは、入力特徴の 1 つの閾値に基づいてデータの分割が実行されます。通常、閾値より小さい場合は左ノードが選択され、そうでない場合は右ノードが選択されます。ノードを通過していくに従い、データに対してますます分割が行われ、このプロセスは特定のリーフで停止し結果が得られます。各リーフはクラス·ラベルに関連付けられており、ルートからリーフへのパスによって分類ルールが生成されます。

3.1 各種のデシジョン·ツリー

デシジョン·ツリー分類器を生成するためのさまざまなアルゴリズムがあります。

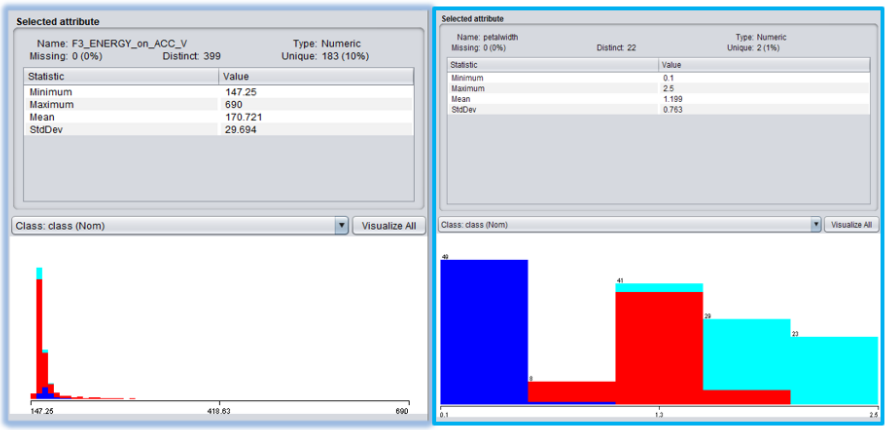
異なるアルゴリズム間の概念的な差異は、ノードでの分割が行われる方法だけではなく、特徴量の重要性、剪定の基準、および停止基準においても異なります。デシジョン·ツリーのトレーニングの基本的なステップは次のとおりです。

a. 分割(ノード): ジニ係数、情報ゲイン、エントロピーなどの分割基準により、特徴量の 1 つを使用して情報を最大化(クラス分離を最大化)することができます。このプロセスでは、最適な特徴量および対応する閾値の選択を行います。図 14 に 2 つの例を示しています。この図は、さまざまなクラスについて特徴量の分布を示しています。この特徴量と適切な閾値を選択した場合、右側の例と比較して、左側の例ではクラスが簡単には分離されないことがわかります。ジニ係数、エントロピー、情報ゲインなどの基準は、特定の閾値に対するクラス間の分離を反映します。
b. 剪定: デシジョン·ツリーの剪定のプロセスでは、最終的なデシジョン·ツリーにおいて特定のノードを削除し、サイズを縮小して過学習を回避します。セクション 4.2.4 で剪定の手法の一例を示します。
c. 停止基準: デシジョン·ツリーは、100%の精度を達成するために深さが増す可能性がありますが、それは過学習の問題につながります。すべてのアルゴリズムは、ノードのさらなる分割を停止する際にさまざまな基準を使用します。この基準は、各ノードにおけるサンプルの最小数や、さらなる分割のための最小ゲインに設定することができます。次の表は、さまざまなタイプのデシジョン·ツリー·アルゴリズムの全体的な差異をまとめたものです。

図13. 各種のデシジョン·ツリー·アルゴリズム

Algo	Splitting criterion	Pruning criterion	Other features
CART	• Gini • Twoing	Cross-validation post-pruning	1. Regression/Classification 2. Nominal/ numeric attributes 3. Missing values 4. Oblique splits 5. Nominal splits grouping
ID3	Information Gain	Pre-pruning	1. Classification 2. Nominal attributes
C4.5	• Information Gain • Information Gain Ratio	Statistical based post-pruning	1. Classification 2. Nominal/numeric attributes 3. Missing values 4. Rule generator 5. Multiple nodes split

図14. さまざまなクラスに関する特徴量の分布

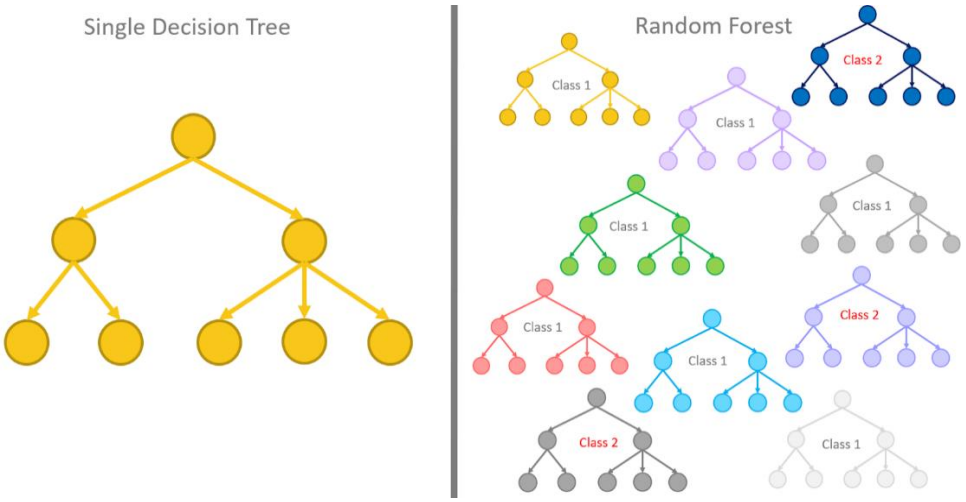


各デシジョン·ツリーのトレーニング·アルゴリズムには特定の利点がありますが、C4.5 はこのアルゴリズムの最新のものであり、データセット内の外れ値に対して堅牢であり、バランスの取れたデシジョン·ツリーを生成します。

3.2 デシジョン·ツリーとランダム·フォレストの比較

ランダム·フォレストはもう 1 つの分類方式です。多数のデシジョン·ツリーは、さまざまな基準を使用して生成されるため、単一のデシジョン·ツリーよりも正確な予測を生成できます。言い換えると、ランダム·フォレストは、わずかに異なる方法でトレーニングされた K（事前に選択されたツリーの最大数）個のデシジョン·ツリーを構築し、それらをマージすることにより、単一のデシジョン·ツリーに比べてより正確で安定した予測を取得します。次の図は、その概念を示しています。

図15. 単一のデシジョン·ツリーとランダム·フォレストの比較



通常、データセットの 70%をトレーニング·データとして使用し、残り（30%）をテスト·データとして使用します。ランダム·フォレストの実装の 1 つは、次のように説明されます。ランダム·フォレストで使用される各デシジョン·ツリーに対して、ブースティングまたはインプレース·ランダム·サンプリングを使用してデータセットとさまざまな特徴量セットを選択し、異なるデシジョン·ツリーを取得し

ます。最終的な出力はさまざまなデシジョン·ツリーの出力の平均になります。したがって、ランダム·フォレストは過学習の影響を受けず、ランダム·フォレストを使用すると、より優れた、より正確かつ安定した予測を得ることができます。MLC では、最大 8 つのデシジョン·ツリーを同時に実行して同一の問題を分類できるため、同様な動作を再現できます。そして、これらのツリーからの出力を処理できるマイクロコントローラ上で結果をマージすることができます。

3.3 デシジョン·ツリーの構築と視覚化

Python では、トレーニング後にデシジョン·ツリーを視覚化できます。下記のコードは、以下を含むプロセスのステップを実装するために使用できるコードです。

- トレーニング·データとテスト·データへのデータ分割
- デシジョン·ツリーの構築
- ツリーの視覚化

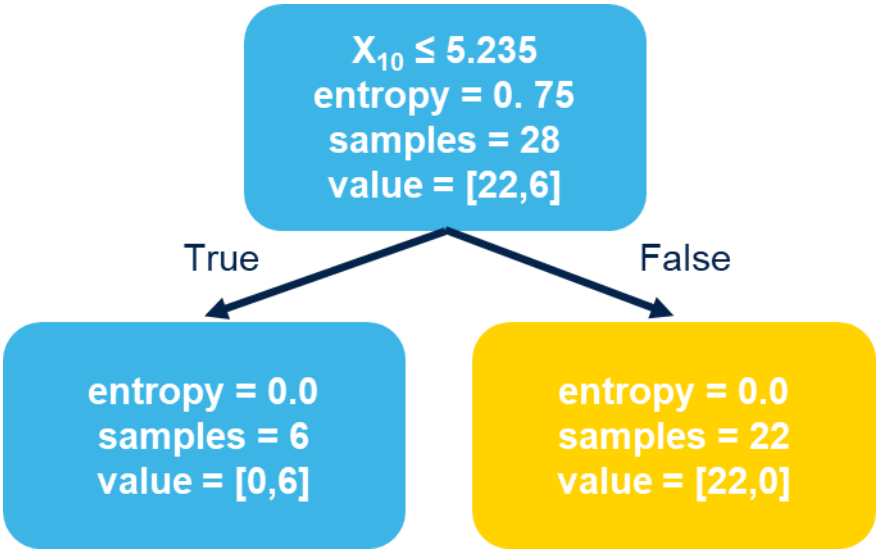
```
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.externals.six import StringIO
from sklearn.tree import export_graphviz

X_train, X_test, y_train, y_test = train_test_split(
    segm, labels, test_size = 0.3, random_state = 101)

clf_entropy = DecisionTreeClassifier(criterion =
    "entropy", random_state = 100)
dot_data = StringIO()
export_graphviz(clf_entropy, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True)
graph = pydotplus.graph_from_dot_data(
    dot_data.getvalue())
```

これは、ノードが 1 つとリーフが 2 つしかない非常に単純な例です。上記のコードは、図 16 に示すデシジョン·ツリーのグラフィック表現を生成します。

図16. Python におけるデシジョン·ツリーのプロットの例



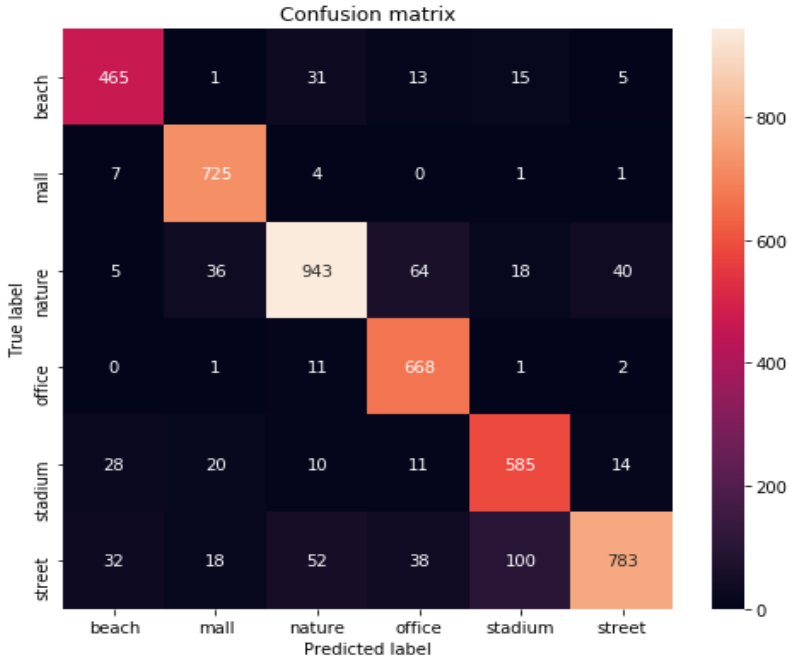
ノードの最初の行は、データを分割するための特定の特徴量に関する条件です。エントロピーに関する次の行は、応答（クラス）の変化です。クラスが 1 つしかない場合、エントロピーのフィールドはゼロになります。サンプルのフィールドは、特定のノードまたはリーフに到達するサンプルの合計です。値のフィールドには、個別のクラスのサンプルに関する情報が含まれています。各分割後のエントロピーが減少し、その減少が大きいほど、ノードの重要性は高くなることに注意してください。

デシジョン·ツリーが小さい場合、視覚化によってノードのルール/条件を理解することが可能となる場合があります。場合によっては、ルールが期待どおりに設定されているかどうかを把握できることがあります。

3.4 予測と混同行列の作成

混同行列は、デシジョン·ツリーの分類性能を要約して表示するために使用されます。この行列は、複数のクラス間の正しい分類と混同の概要を示します。分類アルゴリズムの精度を表すためにさまざまな定義が存在する可能性があるため、混同行列を使用する手法は、「精度」についての誤解を避けるための最も簡単な方法です。次の図に示す例を取り上げてみましょう。

図17.　混同行列によるオーディオ空間環境の分類



縦軸を真のクラスとして設定し、横軸を予測クラスとして設定します。特定のクラスについて、分類アルゴリズムの出力は、対応する行の各セルに表示されます。これにより、真値と予測結果の関係を表すデシジョン·ツリー·アルゴリズム出力のマッピングが生成されます。ここでは、アルゴリズムが「ストリート」クラスと「スタジアム」クラスの間で混同され易くなっていることがわかります。混同行列における最終行の 5 番目のセルは、アルゴリズムが本来「ストリート」クラスの中の 100 サンプルを「スタジアム」クラスに誤って分類したことを示しています。次の Python のコードを使用すると、図 17 に示すように混同行列を構築できます。

```
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
from sklearn.externals.six import StringIO
from sklearn.tree import export_graphviz
from sklearn.metrics import confusion_matrix

def confusion_matrix(y_test, y_pred, class_list):
    confusion_matrix = metrics.confusion_matrix(y_test,
                y_pred)
    fig_confuse = plt.figure(figsize=(4, 3.5))
          sns.heatmap(confusion_matrix, xticklabels=
          sorted(class_list), yticklabels = sorted(class_li
          st), annot=True, fmt="d");
    plt.title("Confusion matrix")
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.show();

class_list = ["beach", "mall", "nature", "office",
                    "stadiu  m", "street"]
y_pred_entropy = clf_entropy.predict(X_test)
confusion_matrix(y_test, y_pred_entropy, class_list)
```

4. モデルの評価

トレーニングしたデシジョン·ツリーのモデルが十分に高性能であることをどうすれば確認できるでしょうか。モデルの性能を測定するために利用できる手法は複数あります。第 1 に、テスト·データセットを使用して混同行列を確認する必要があります。安定したモデルを選択するためには、K 分割（交差検証）を実装することができます。次のセクションでは、モデルの品質を判断する方法と、モデルの問題を修正する方法について説明します。

交差検証は、アウト·オブ·サンプル手法としても知られています。モデルのトレーニングとテストを行うために、データをテストとトレーニングへランダムに分割することがよくあります。トレーニング·データは多様なデータを取り込んだものと想定され、モデルのトレーニングには十分です。交差検証を使用する具体的な理由は以下の 3 つです。

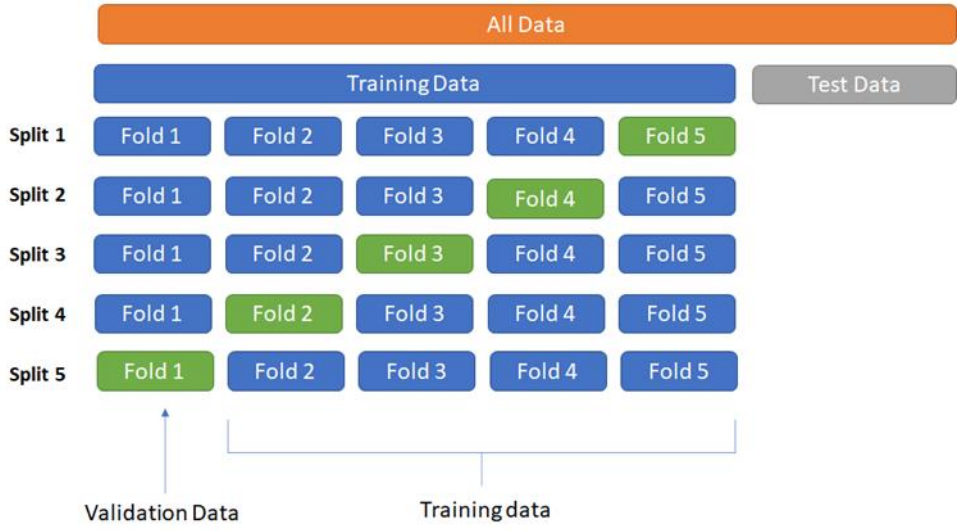
- まず、いくつかの特定の問題について、ランダム·サンプリング方式では、テスト·データには含まれるが、トレーニング·データには含まれないない特定のユース·ケースのすべてのデータを選択する場合があります。この場合、モデルはその特定のユース·ケース向けにトレーニングされていないため、モデルの性能は低下します。データの変動が周期的であり、ランダム·サンプリングでデータを分割してトレーニングおよびテスト·データの一部にすることができない場合も考えられます。
- 第 2 に、ランダム分割を実行するため、モデルの精度はランダム分割のたびに変化し続けます。
- 第 3 に、検証で使用されるデータは、モデルの検証と選択にのみ使用されます。モードを選択したら、このデータを使用してモデルの精度を向上させることはできません。

交差検証には利用できるさまざまな手法があります。そのそれぞれに特定の長所と短所があります。標準的な交差検証の手法は次のとおりです。

- 1 個抜き
- p 個抜き
- 反復ランダム·サブサンプリング
- K 分割
- ホールドアウト

最もよく使用される手法は、K 分割交差検証（具体的には 10 分割）です。K 分割交差検証では、トレーニング·データを k 個に分割する必要があり、次の図 18 に示すように、トレーニングは k 回実行されます。

図18.　5 分割の交差検証



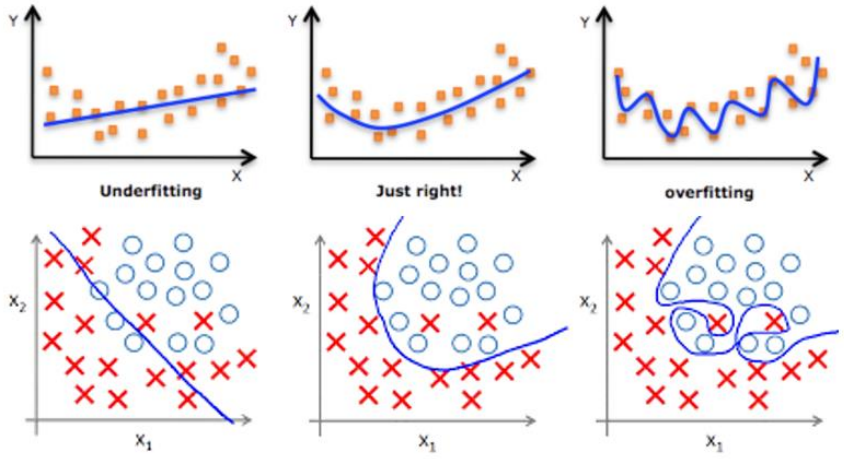
サブセットの 1 つは検証セットとして使用され、その他の k-1 個のサブセットはトレーニング·セットとして使用されます。このようにして、各サブセットはトレーニングに k-1 回、検証に 1 回使用されます。この手法の利点は、誤差の推定が k 回の試行すべてにわたって平均化され、モデルの総合的な有効性が得られることです。K 分割検証の最後には、すべてのトレーニング·データを利用して最終的なモデルが作成されます。

K 分割交差検証を実行すると、その検証は、平均二乗誤差、二乗平均平方根誤差、または絶対中央値偏差などの統計的尺度を通じてさまざまな誤差を集計します。これらのパラメータは、モデルの品質（学習不足、過学習など）を確認する際に適切なモデルを選択する上で役立ちます。次のサブセクションでは、モデルの品質について詳しく説明します。

4.1　学習不足

学習不足は、機械学習モデルが十分に複雑でない場合や、データセットに含まれる関係と情報を取得するための十分な情報（特徴量）がない場合に発生します。学習不足、適性学習（ロバスト）、および過学習の概念を図 19 に示します。

図19.　学習不足、適性学習（ロバスト）、過学習の概念



モデルが学習不足でないかどうかを判定するには、どうすればよいでしょうか。モデルが学習不足であれば、トレーニング·データに含まれるデータやパターンを捉えることができません。言い換えると、トレーニングされたモデルがトレーニング·データとテスト·データの両方似たいして性能が低い場合は、学習不足が原因である可能性があります。このシナリオでは、選択された特徴量は、モデルがデータに適合するための情報を取得する上で十分でない可能性があります。

学習不足は対処しやすく、標準的な解決策は、特徴量を追加してモデルを再びトレーニングすることです。もう 1 つの選択肢は、クラスの数を減らすことです。場合によっては、2 つのクラス間のデータが類似しているように見える場合、モデルはそれを分類する際に適切な処理をすることができない可能性があります。多様なデータを追加することも、学習不足の問題に対処する上で役立つ場合があります。モデルは複雑さを増し、前のトレーニングで見逃されていたパターンを学習できる可能性があるからです。

4.2　過学習

学習不足とは異なり、過学習は、モデルがトレーニングされたデータに対しては高性能である一方、未知のデータに対しては低性能である場合に発生するモデリングのエラーです。トレーニング·データセットでは精度は高くなりますが、テストまたは新たに収集された別のデータセットに対しては精度が低くなります。過学習のモデルは、通常、トレーニングで使用されるデータの特徴を取り込むために、過度に複雑なモデルを作成するという形を取ります。この状況は、ノイズを実際のパターンとして解釈することにより、モデルができるだけ多くのポイントに適合する上で大幅な柔軟性と自由度を持っている場合に発生します。K 分割検証または交差検証は、一般に過学習を識別する上で役立ちます。トレーニング·データセットでモデルを実行すると、95%以上の精度で予測する一方、テスト·データでは 50～70%の精度を示す場合、これは過学習問題の適切な識別になります。K 分割検証の RMSE（二乗平均平方根誤差）は、検証データのさまざまな分割に関する累積誤差です。最終的なモデルの精度は良好である一方、RMSE が高い（>0.2）場合は、過学習の可能性が高いと考えられます。

以下の手法は、過学習に対処する上で有効です。

4.2.1 さらなるログの収集

これは通常、過学習の問題に対する簡単な解決策です。新しいログは、より多様性のある別の状況で収集する必要があります。収集されるデータは、モデルが実世界のアプリケーションで遭遇するさまざまなユース·ケースを取り込む必要があります。

4.2.2 データ拡張

ログを収集するユーザが複数いない場合や、多様性のある環境/条件を設定することが容易でない場合もあります。ノイズを追加したり、データをさまざまな角度で回転させたりすることを検討できます。頭部ジェスチャ検出プロジェクトの例を取り上げましょう。このプロジェクトの目的は、クラス「うなずく」、「揺らす」、「静止」、「歩く」、「振る」を検出することであり、センサはヘッドセットに取り付けられています。y 軸は地球（重力と同じ方向）を指し、x 軸はユーザの進行方向であると予想されます。

しかし、トレーニング·データに限られた設置方向によるデータや限られた数のユーザによるデータしかない場合、モデルには 2 つの問題が発生する可能性があります。まず、このデータでトレーニングされたデシジョン·ツリーは、ログを収集する人々に合わせて調整され、他の未知のユーザに対しては低性能を示すと考えられます。次に、モデルは特定の設置方向に対して調整されており、性能はヘッドセットの装着方法対するユーザの好みによって影響を受ける可能性があります。これらの問題を克服するためには、次のことが可能です。

- 前述のようにノイズを注入し、データの小さな変動に対処する。
- 既存のデータを（使用中に）可能なさまざまな角度で回転させ、この回転したデータをトレーニング·データセットに追加する。トレーニングされたモデルは、より多くのユーザの好みのほとんどをカバーし、ソリューションはセンサの向きに対して十分に堅牢となる。

4.2.3 特徴量の数の削減

トレーニング作業時に多数の特徴量を選択すると、モデルをすべてのトレーニング·データに適合させるための柔軟性と自由度を拡大できますが、過学習を引き起こす可能性もあります。あまり重要でない特徴量を削除すると、モデルの複雑さが低減され、過学習の問題を解決する上で役立ちます。特徴量の可視化のセクションで説明したように、特徴量をプロットして可視化すると、特徴量とクラスの関係を理解する上で有効です。特徴量が意味をなさない場合、ユーザはモデルから特徴量を削除して複雑さを軽減できます。

4.2.4 剪定

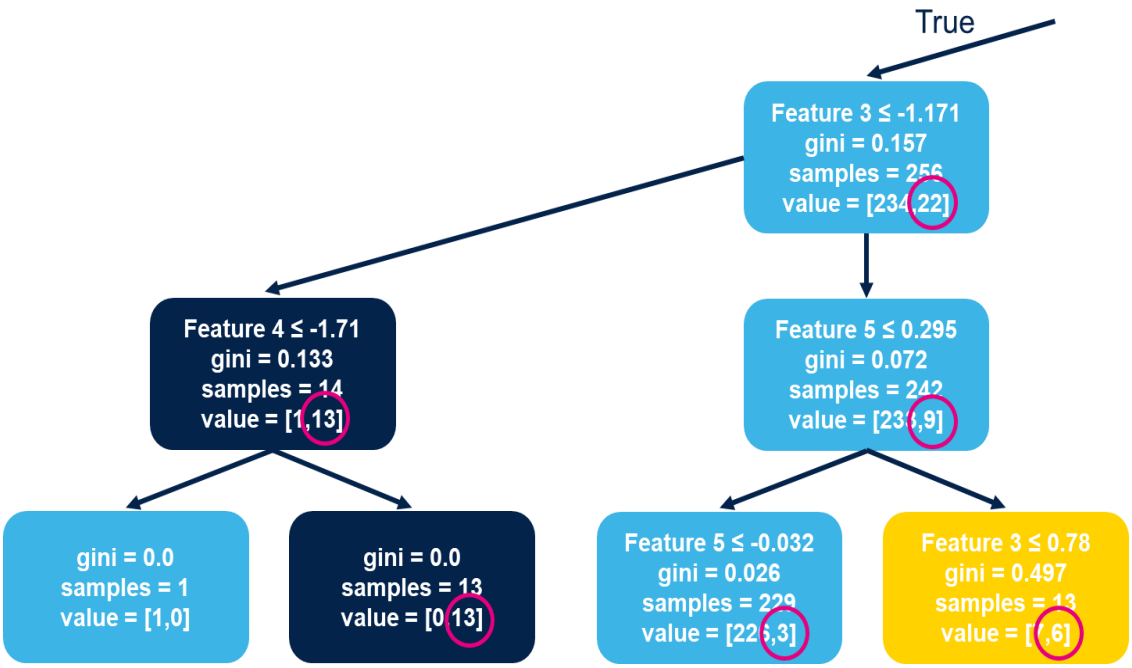
剪定とは、重要でないツリーのセクションを除去することにより、デシジョン·ツリーを単純化することです。これは、あまり多くのデータ·ポイントを分類しないノードに対応する可能性があります。これは、モデルの成長に対する早期停止ポイントと考えることができます。このポイントでノードの分岐を停止することになります。さまざまなタイプのデシジョン·ツリーには、図 13 に示すように、ツリーを剪定するためのさまざまな基準があります。剪定には次の 2 つのタイプがあります。

- 事前剪定：事前剪定は、サブセットの分類を続行する前に、ツリーの成長を停止します。
- 事後剪定：デシジョン·ツリーを構築した後、ロジックによって特定のノードまたはサブツリーのトリミングについて決定します。

ほとんどの場合、ツリーの成長を停止する正確な瞬間を知ることは容易ではないため、事後剪定が選択されます。事後剪定では、ロジックはまず完全なデシジョン·ツリーを成長させ、次にある基準（情報ゲイン、クラスの不均衡など）を使用して、デシジョン·ツリーの重要でない部分の除去を開始します。ここで、ユーザが事後剪定について理解するための実用的な例を示します。特定のクラスの少数のサンプルのみを分類する特定のノードがデシジョン·ツリーにあり、残りの多数のサンプルが他のクラスに属する場合は、ノードを「他の」クラスに属するすべてのサンプルを分類するリーフに置き換えます。得られるデシジョン·ツリーの性能は、この操作によって悪影響を受けることはありません。

剪定は、過学習を引き起こす可能性があるデシジョン·ツリーの一部であるリーフを削減します。そのため、剪定は過学習の削減に役立ち、ソリューションを堅牢にすることができます。次の図 20 に示すノードを例として取り上げましょう。

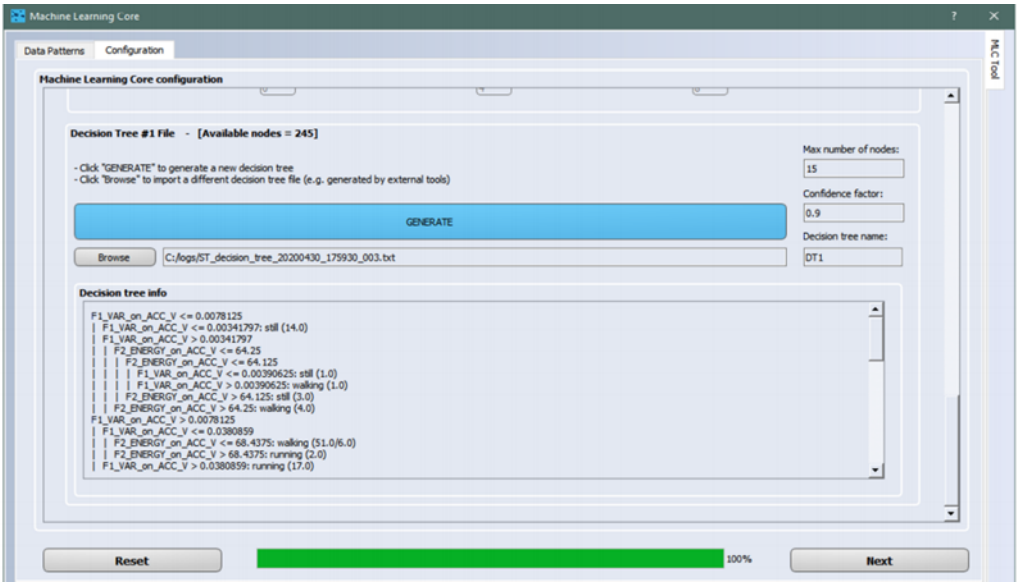
図20. プルーニングの例



上記のデシジョン·ツリーの左側のノード（Features 4 ≤ -1.71）は、1 つの例外サンプルのみ分類しますが、他の 13 個のサンプルはすべて別のクラスに属します。この場合、このノード剪定しリーフとする方がよいかもしれません。

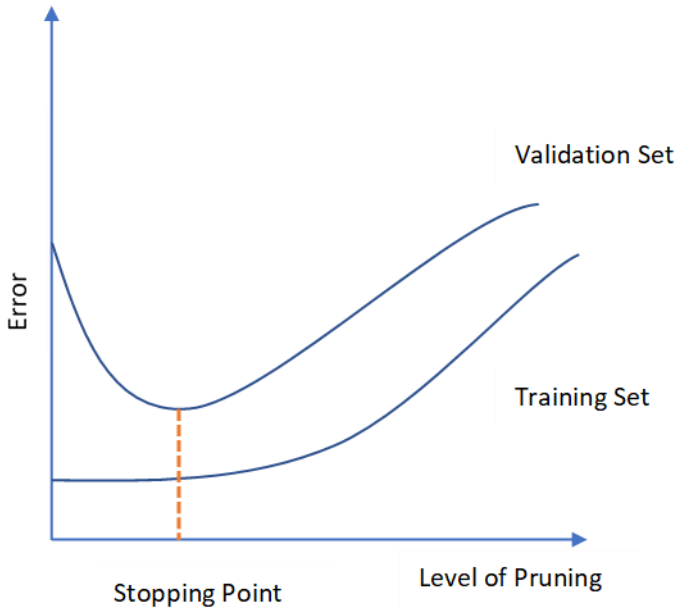
Unico-GUI 内のデシジョン·ツリー·アルゴリズムには、剪定のオプションがあります。限られた数のノードを設定でき、条件が満たされるまで確信度が切り詰められます。

図21. Unico-GUI における MLC のデフォルトのデシジョン·ツリー·アルゴリズム



最適な剪定を決定するための手法は、検証とトレーニングの精度を監視することで実現されます。検証データセットとトレーニング·データセットの両方の精度/誤差を剪定のレベルに対してプロットすると、通常、下の図 22 に示すようなプロットが得られます。

図22.　プルーニングの停止ポイント



剪定しない場合、トレーニング·データに対するモデルの誤差は最小になりますが、過学習のために検証セットに対する誤差が大きくなる可能性があります。剪定のレベルを引き上げると、トレーニング·セットにおける誤差は増加しますが、一部のノードがノイズに過学習されるため、検証セットに対する誤差は減少し始めます。剪定をさらに増すと、剪定によって正しく分類しているノードが除去されるため、検証セットとトレーニング·セットの両方で誤差が増加します。これは剪定プロセスを停止するポイントです。

4.3　メタ分類器

メタ分類器は、機械学習モデルに精度、特に計算されたモデルからの出力の安定性を向上させるための追加的な方法を提供します。メタ分類器は、(デシジョン·ツリーの形式に制約されない)分類アルゴリズムに適用でき、デシジョン·ツリーの出力に対する平滑化フィルタ(Maximum Votingなど)として機能します。それぞれの MLC のアプリケーション·ノートにメタ分類器の詳細な説明があります。

次の例は、クラス間の移行が頻繁ではないユース·ケースにメタ分類器を使用する利点を示しています。2 つのクラス「目覚め」と「睡眠中」について、スマートウォッチにセンサが取り付けられている場合の活動検出の事例を考えてみましょう。手首がわずかに動くと、ユーザが実際には「睡眠中」の時に分類出力が「目覚め」になる可能性があります。メタ分類器は、分類アルゴリズムからの出力を平滑化し、出力から偽の「目覚め」分類を除去することができます。

参考資料

関連資料

製品番号	市場	ファミリ	アプリケーション·ノート
LSM6DSOX	民生用	iNEMO 慣性モジュール（IMU）	AN5259
LSM6DSRX	民生用	iNEMO 慣性モジュール（IMU）	AN5393
ISM330DHCX	産業用	iNEMO 慣性モジュール（IMU）	AN5392
IIS2ICLX	産業用	加速度センサ	AN5536

機械学習関連のリソース

MLC 向け GitHub プロジェクト	**https://github.com/STMicroelectronics/STMems_Machine_Learning_Core**
ST の MLC エコシステム	https://www.st.com/content/st_com/ja/ecosystems/MEMS-Sensors-Ecosystem-for-I Learning.html
ST の ML 向け MEMS コミュニテ	**community.st.com/s/group/CollaborationGroup**
MEMS とセンサに関する Q&A	**Mems-and-Sensors Q&A**

改訂履歴

日付	バージョン	変更点
2020 年 11 月 3 日	1	初版発行
2021 年 2 月 8 日	2	文書全体にわたる改善。図 4 を追加、図 16、図 20 を更新、「機械学習関連のリソース」を追加

日付	バージョン	変更点
2021 年 7 月 1 日	2	日本語版　初版発行