第2章 データ形式をINT8に量子化したオートエンコーダのモデルを準備する

ステップ②：マイコンでも動くように学習済みモデルを軽量化

関本 健太郎

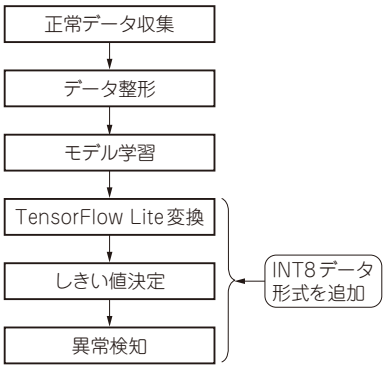

図1 改造するプログラムは3つ

前章で作成したオートエンコーダ・モデルは，入出力が32ビット浮動小数点（FP32）であり，推論の実行にはデスクトップPCやラズベリー・パイ5などの比較的高性能な環境を必要とします．マイコン・ボード上でモデル推論を行うには，推論計算量を大幅に削減する必要があります．その手法として，モデルの入出力および内部データにINT8形式（またはUINT8形式）を用いる量子化[注1]があります．

本章では，前章の手順を拡張し，INT8量子化への対応を行います（**図1**）．

改造1…モデル学習プログラム train_autoencorder.py

● 入力サンプルの準備

リスト1に示すtrain_autoencoder.pyは，学習済みのKerasオートエンコーダをフル整数（INT8：入出力ともにint8）のTensorFlow Lite（以降，TF Lite）モデルへ変換/保存する処理です．

まず，rep_ds()で代表データを定義します．代表データには，学習で用いたZスコア空間の配列X[注2]をそのまま用い，念のため－K_CLIP，K_CLIPにクリップ[注3]した上で，形状（1，－1）の1サンプル[注4]として最大2048件を供給します．これにより，実運用時の分布に近いスケールとゼロ点（量子化パラメータ）が推定されます．

リスト1　モデル学習プログラムtrain_autoencoder.py（抜粋）

```
: （省略）
    # ===== 新規: TFLite INT8（入出力ともにint8）を追加出力 =====
    def rep_ds():
        # 代表データは学習時の Z スコア空間（[-K, K] へクリップ）で与える
        for i in range(min(len(X), 2048)):  # 上限で軽量化
            x = X[i].astype(np.float32).copy()
            # 念のためクリップ（学習作成済みXがZスコア済みである前提）
            x = np.clip(x, -K_CLIP, K_CLIP)
            yield [x.reshape(1, -1)]

    converter_i8 = tf.lite.TFLiteConverter.from_keras_model(autoencoder)
    converter_i8.optimizations = [tf.lite.Optimize.DEFAULT]
    converter_i8.representative_dataset = rep_ds
    converter_i8.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
    converter_i8.inference_input_type  = tf.int8
    converter_i8.inference_output_type = tf.int8
    tflite_model_i8 = converter_i8.convert()
    with open("autoencoder_model_int8.tflite", "wb") as f:
        f.write(tflite_model_i8)
    print(" モデルを autoencoder_model_int8.tflite に保存しました。")
```

● モデル変換と利点

TF Liteコンバータを作成し，12行目で最適化を有効化し，13行目で代表データを登録します．さらに，supported_opsをTFLITE_BUILTINS_INT8のみに限定し，inference_input_typeとinference_output_typeをいずれもtf.int8に設定することで，内部演算から入出力まで浮動小数点へ変換しない完全なINT8形式のモデルを強制しま

注1：ニューラル・ネットワークの重み/中間演算/入力/出力を全て8ビット整数（INT8）に変換して推論を行う最適化手法．

注2：各特徴量を平均0/標準偏差1に正規化したデータ配列．

注3：データの値を範囲－K_CLIP，K_CLIPに収め，それ以上や以下の値を切り詰める処理．

注4：1行に全ての特徴量を並べた1サンプル分の2次元配列．