GMM Classifier

Description

The Gaussian Mixture Model Classifier (GMM) is basic but useful supervised learning classification algorithm that can be used to classify a wide variety of N-dimensional signals.

The GMM algorithm is part of the GRT classification modules.

Advantages

The GMM algorithm is a good algorithm to use for the classification of static postures and non-temporal pattern recognition.

Disadvantages

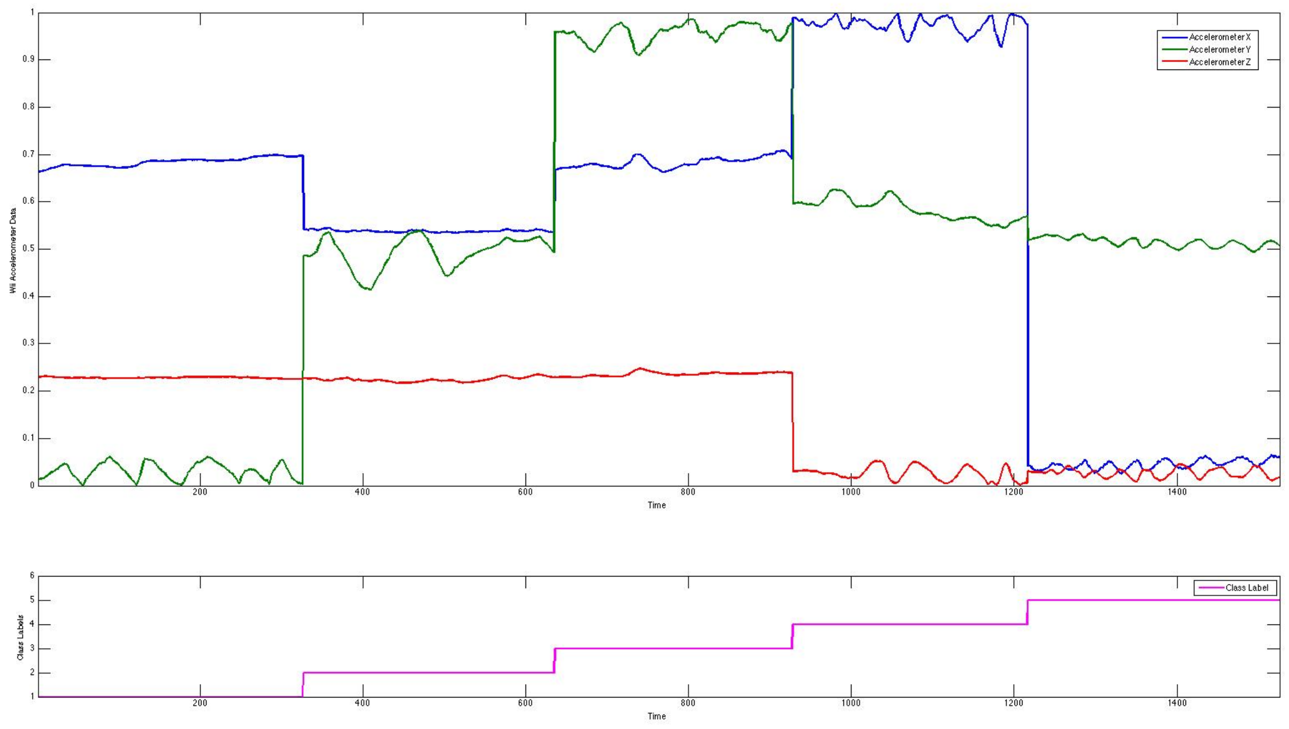
The main limitation of the GMM algorithm is that, for computational reasons, it can fail to work if the dimensionality of the problem is too high (i.e. greater than 6 dimensions for instance). If this is the case with your data then you might want to try either the ANBC or Support Vector Machine classification algorithms instead. Another disadvantage of the GMM algorithm is that the user must set the number of mixture models that the algorithm will try and fit to the training dataset. In many instances the user will not know how many mixture models should be used and may have to experiment with a number of different mixture models in order to find the most suitable number of models that works for their classification problem.

Training Data Format

You should use the ClassificationData data structure to train the GMM classifier.

Example Code

This examples demonstrates how to initialize, train, and use the GMM algorithm for classification. The example loads the data shown in the image below and uses this to train the GMM algorithm. The data is a recording of a Wii-mote being held in 5 different orientations, the top graph shows the raw accelerometer data from the recording (showing the x, y, and z accelerometer data), while the bottom graph shows the label recorded for each sample (you can see the 5 different classes in the label data). You can download the actual dataset in the Code & Resources section below.



The data is a recording of a Wii-mote being held in 5 different orientations, the top graph shows the raw accelerometer data from the recording (showing the x, y, and z accelerometer data), while the bottom graph shows the label recorded for each sample (you can see the 5 different classes in the label data). WiiAccelerometerData.jpg

```
/*
 GRT GMM Example
 This examples demonstrates how to initialize, train, and use the GMM algorithm for classification.

 The Gaussian Mixture Model Classifier (GMM) is basic but useful classification algorithm that can be used to
 classify an N-dimensional signal.

 In this example we create an instance of a GMM classifier and then train the algorithm using some pre-recorded
training data.
 The trained GMM algorithm is then used to predict the class label of some test data.

 This example shows you how to:
 - Create an initialize the GMM algorithm
 - Load some ClassificationData from a file and partition the training data into a training dataset and a test dataset
 - Train the GMM algorithm using the training dataset
 - Test the GMM algorithm using the test dataset
 - Manually compute the accuracy of the classifier
*/

//You might need to set the specific path of the GRT header relative to your project
#include "GRT.h"
using namespace GRT;

int main (int argc, const char * argv[])
{

    //Create a new GMM instance and set the number of mixture models to 2
    GMM gmm;
    gmm.setNumMixtureModels(2);

    //Load some training data to train the classifier
    ClassificationData trainingData;

    if( !trainingData.loadDatasetFromFile("GMMTrainingData.txt") ){
        cout << "Failed to load training data!\n";
        return EXIT_FAILURE;
    }

    //Use 20% of the training dataset to create a test dataset
    ClassificationData testData = trainingData.partition( 80 );

    //Train the classifier
    bool trainSuccess = gmm.train( trainingData );

    if( !trainSuccess ){
        cout << "Failed to train classifier!\n";
        return EXIT_FAILURE;
    }

    //Save the GMM model to a file
    bool saveSuccess = gmm.saveModelToFile("GMMModel.txt");

    if( !saveSuccess ){
        cout << "Failed to save the classifier model!\n";
        return EXIT_FAILURE;
    }

    //Load the GMM model from a file
    bool loadSuccess = gmm.loadModelFromFile("GMMModel.txt");

    if( !loadSuccess ){
        cout << "Failed to load the classifier model!\n";
        return EXIT_FAILURE;
    }

    //Use the test dataset to test the GMM model
    double accuracy = 0;
    for(UINT i=0; i<testData.getNumSamples(); i++){
        //Get the i'th test sample
        UINT classLabel = testData[i].getClassLabel();
        vector< double > inputVector = testData[i].getSample();

        //Perform a prediction using the classifier
        bool predictSuccess = gmm.predict( inputVector );

        if( !predictSuccess ){
            cout << "Failed to perform prediction for test sampel: " << i <<"\n";
            return EXIT_FAILURE;
        }

        //Get the predicted class label
        UINT predictedClassLabel = gmm.getPredictedClassLabel();
        vector< double > classLikelihoods = gmm.getClassLikelihoods();
        vector< double > classDistances = gmm.getClassDistances();

        //Update the accuracy
        if( classLabel == predictedClassLabel ) accuracy++;

        cout << "TestSample: " << i <<  " ClassLabel: " << classLabel << " PredictedClassLabel: " <<
predictedClassLabel << endl;
    }

    cout << "Test Accuracy: " << accuracy/double(testData.getNumSamples())*100.0 << "%" << endl;

    return EXIT_SUCCESS;
}
```

Code & Resources

GMMExample.cpp GMMTrainingData.txt

Documentation

You can find the documentation for this class at GMM documentation .