ECE5242 Project2: Gesture recognition using HMM

Sijia Gao

February 2019

Abstract

The goal of this project is to use IMU sensor readings from gyroscopes and accelerometers to train a set of Hidden Markov Models and recognize different arm motion gestures. To achieve this, we first train a Hidden Markov Model for each training sequence in each gesture class using Baum-Welch algorithm. The testing sequence is then classified to the class of HMM with highest likelihood.

1 Introduction

Compared with the classification of a static object such as the red barrel in RGB images in project1, we aim to classify "moving" objects in terms of its dynamics over an interval. Put it more into real life, say we have a video capturing a girl eating a cake. The dynamic of the object (girl) consists of a sequence of actions (states): raising hand, approaching the cake, putting the cake into her mouth which evolves according to a Markov chain (with left to right transition matrix). The problem is that given the video, how can we justify this video capturing a girl eating a cake. The solution to this problem is to learn a Hidden Markov Model (HMM) from the video. "Hidden" means the states (raising hand, approaching the cake and putting the cake into mouth) are unknown but only measurement (video) is available.

The organization of this report is as follows: Sec.2 describes the project setup; Sec.3 gives details of the algorithm used to learn HMM model; Sec.4 illustrates performance of the proposed method on the test set. Some alternative approaches are mentioned in Sec.5 and Sec.6 concludes the whole report.

2 Problem Formulation

In this project, we aim to classify a sequence of IMU sensor recordings from gyroscopes and accelerometers to one of the 6 gesture classes: beat3, beat4, circle, eight, inf,wave. These 6 gesture classes are specified in the given training dataset where each class contains several sequences of IMU sensor recordings.

The general idea of the classifier proposed in this report is that we learn a HMM for each training sequence in each class; then classify the testing sequence in terms of maximum likelihood. The pipeline of our method is illustrated as follows:

1. Filter IMU sensor recording by a median filter (this step is not included in the initial code submission)
2. Specify size of state space $|\mathcal{X}|$ and observation space $|\mathcal{Y}|$. Quantize training sequences using Kmeans with number of clusters= $|\mathcal{Y}|$
3. Use Baum-Welch algorithm to learn a HMM for each training sequence in each class. Denote λ_{ij} the trained model for the jth training sequence in gesture class i.
4. Given trained models in step3 and the testing sequence $y_{1:T}^{\text{test}}$, compute likelihood $p(y_{1:T}^{\text{test}}|\lambda_{ij})$. Assign testing sequence the class of HMM with the maximum likelihood: $i^* = \underset{\forall i,j}{\text{argmax}} p(y_{1:T}^{\text{test}}|\lambda_{ij})$.

3 Baum-Welch algorithm

HMM={A,B} is a 2 tuple structure that consists of transition matrix A and observation matrix B. The dynamics of a HMM can be written as

$$\begin{aligned} x_k &\sim A(x_{k-1}) \\ y_k &\sim B(x_k) \end{aligned} \tag{1}$$

The first, second line in (1) is called state equation and observation (measurement) equation, respectively. The state sequence $x_1, x_2, \ldots, x_T$ with $x_k \in \mathcal{X}$ is hidden and evolves according to the transition matrix $A : \mathcal{X} \to \mathcal{X}$. Given the observation matrix $B : \mathcal{X} \to \mathcal{Y}$,the output of HMM is the observation sequence $y_1, y_2, \ldots, y_T$ with $y_k \in \mathcal{Y}$. Note that y_k is only dependent on x_k.

Baum-Welch algorithm is EM(expectation maximization) algorithm to estimate $\{A, B\}$ given observation sequence $y_{1:T}$. Baum-Welch algorithm operates in two steps:

1. E step: compute an auxiliary likelihood which requires a forward-backward algorithm
2. M step: update parameters $\{A, B\}$

Baum-Welch algorithm is summarized in Algorithm1. We implement Algorithm1 for several iterations until $\{A, B\}$ converge or exceed a maximum number of iterations. The likelihood of a model λ_{ij} given the testing sequence is computed as

$$\log(p(y_{1:T}^{\text{test}}|\lambda_{ij})) = -\sum_{k=1}^{T} \log(c(k)) \tag{2}$$

where $c(k)$ is computed in Algorithm1.

```
**E step (forward-backward)**:
**input**$\{A, B\}$
initialize $\alpha_1(i) = \pi(i) * b_i(y_1)$,$\pi(i) = \frac{1}{|\mathcal{X}|}$
**for** $k = 2 : T$ **do**
    $\alpha_k(i) = \sum_{j=1}^{|\mathcal{X}|} \alpha_{k-1}(j) a_{ji} b_i(y_k)$
    $c(k) = 1/\sum_{i=1}^{\mathcal{X}} \alpha_k(i)$
    $\alpha_k(i) = \alpha_k(i) c(k)$
**end**
initialize $\beta_T = \mathbf{1}$
**for** $k = T - 1 : 1$ **do**
    $\beta_k(i) = \sum_{j=1}^{\mathcal{X}} \beta_{k+1}(j) a_{ij} b_j(y_{k+1})$
    $\beta_k(i) = \beta_k(i) c(k)$
**end**
**for** $k = 1 : T$ **do**
    $\gamma_k(i) = \alpha_k(i) \beta_k(i)$
**end**
**for** $k = 1 : T$ **do**
    $\gamma_k(i, j) = \alpha_k(i) a_{ij} b_j(y_{k+1}) \beta_{k+1}(j)$
**end**
**M step**:
$a_{ij} = \frac{\sum_{k=1}^{T} \gamma_k(i,j)}{\sum_{k=1}^{T} \gamma_k(i)}$
$b_{ij} = \frac{\sum_{k=1}^{T} \gamma_k(i) 1(y_k=j)}{\sum_{k=1}^{T} \gamma_k(i)}$
**return** $\{A, B\}$
```

Algorithm 1: Baum-Welch algorithm

Table 1: Performance on 2019Proj2-train-additional

filename	classifier
beat3-31	beat3
beat4-31	beat4
wave31	wave
circle31	circle
inf31	inf
eight31	eight

Table 2: Performance on 2019Proj2-test

filename	groundtruth	classifier
test1	wave	wave
test2	beat4	beat4
test3	inf	inf
test4	beat3	beat3
test5	circle	circle
test6	inf	inf
test7	eight	eight
test8	beat4	beat4

4 Performance

We test the proposed classifier using two data sets: 2019Proj2-train-additional and 2019Proj2-test. We choose median filter windowsize=3, size of state space $|\mathcal{X}| = 8$, size of observation space $\mathcal{Y} = 10$ and set maximum number of iterations=20. Please be careful with window size in median filter as large window size may lead to poor classification results. The performance of the proposed classifier on the two datasets are illustrated in Table.1 and Table.2. Results show 100% classification accuracy on both two testing data sets. Results in initial code submission are shown in Table.3 with 2 false classifications on “beat3” and ”beat4”.

Table 3: Old Performance on 2019Proj2-test

filename	groundtruth	classifier
test1	wave	wave
test2	beat4	beat3
test3	inf	inf
test4	beat3	beat3
test5	circle	circle
test6	inf	inf
test7	eight	eight
test8	beat4	beat3

5 Alternative approaches and discussion

One may construct a HMM model for each gesture class (This is the approach we implemented in the initial code submission). In detail, for a gesture class we concatenate all its training sequence and train a HMM model. However, the resulted classifier does not perform well between “beat3” and “beat4”. One possible explanation could be this method results in too few “templates” for the testing sequence to match. Therefore, we train each training sequence for each gesture class a HMM and achieve improvement. Another potential approach could be train a continuous HMM instead of a discrete HMM. In other words, the observation equation in (1) under this case is

$$y_k = x_k + w_k \tag{3}$$

where w_k can be white Gaussian noise. However, the resulted classifier is sensitive to parameter initialization, e.g., mean and covariance matrix of w_k. A possible explanation for this could be x_k, which is a six-dimension data consisting of gyroscope and accelerometer is at different scales among dimensions. Classification between ”beat3” and ”beat4” is tricky, transformation of IMU sensor recording into other representation space such as orientation does not give much improvement. Therefore, we seek other trials such as median filter for preprocessing and results in improvement in classification accuracy.

6 Conclusion

In this report, we established HMM models to classify the gesture of a sequence of IMU sensor recordings. The classifier we proposed includes three steps. First, we quantized IMU sensor recordings using Kmeans. Second, we constructed a HMM for each training sequence for each class. We utilized the well-known Belm-Welch algorithm to estimate HMM parameters including transition and observation matrix. Finally, given a testing sequence, we computed the likelihood for each trained HMM model. The testing sequence was assigned the class of HMM with highest likelihood.