ESE 650, Learning in Robotics, Spring 2016: Project3

Yu-Cheng Lin

Introduction

This project aims to identify gestures from IMU measurements using hidden Markov models. This model works for this project because of the richness in hidden states. Also, a Markov process naturally works for a series of measurement in time.

Training Implmentation

Even though the implementation model is fairly straight forward, there are some nuances that I must look out for in order to train a reasonable model.

0.1 Model Choice

HMM comes in many different flavors. For this project specifically, I chose the bare minimum of the implementation: possible transition at every time step and discrete observation. I also counted each training instances as separate entities, i.e. each training instance has independent forward probability α and backward probability β. In EM, the sum of transition likelihood at a specific time step ξ_t and its normalizing counterpart γ_t are the cumulative sum across all the training examples. I started with this simple model because it is the easiest to implement and it is easy to vectorize in MATLAB. With enough cross validation and training, this model actually worked out to be good enough for gesture recognition.

0.2 Avoiding Underflow

Since HMM is a chained linear system, it is obvious that the probability of observing some output exponentially decays, and this leads to numerical underflow very quickly. To solve this problem, at every step of forward and backward propagation, I pull out the magnitude of the alpha and store it separately.

$$\alpha_1 = C_1 \hat{\alpha_1}$$

where $C_1 = |\alpha_t|_1$
The sequential update for C_{t+1} is then

$$C_{t+1} = C_t |\alpha_t|_1$$

Note this still underflows, but now we can take the log of C_t and store it, and the sequential update will be the sum of the logs.

It is interesting that because of we are normalizing the sums of xi and $gamma$, the magnitude drops out, even in the multiple training example case, rendering it useless in training. This log magnitude is useful in the final recognition since I need to evaluate the probability of a sequence given the model.

0.3 Local Maxima

HMM does not have one global maxima because of the latent variables (states). One way to find a good model via MLE is EM. However, EM frequently gets stuck in a local maxima. To find better local maxima, each model is trained multiple times with different random initial conditions. And the best model (highest joint probability across all the training examples) is chosen.

0.4 Cross Validation

Due to lack of number in the training examples, I can easily do LOOCV. It turns out that the recognition is pretty robust to model complexity. The motions must be pretty different. I chose a model with 4 latent states and 5 output clusters in the end just because I liked these numbers.

Test Set Performance