ADVANCED MACHINE LEARNING :: ASSIGNMENT 1
Hidden Markov Models For Recognizing Human Activity

README
Girish Rao
gr2183@columbia.edu
10/10/2006

Files:
hmm_train.m
hmm.m
hmm_main.m
hmm_test.m
animate.m

hw1.mat
trainrun.mat
varW.mat (generated by train_hmm.m)
varR.mat (generated by train_hmm.m)

walk.pdf
run.pdf

Running the programs:
First,
```
>> load hw1.mat
```
Then train all sequences. This is done by simply running hmm_train once
```
>> hmm_train
```
Then, to test on a specific sequence, run hmm_main, for example,
```
>> hmm_main(testr, 3)
```

After running hmm_train program, trainrun.mat, varW.mat and varR.mat should be generated by the program and automatically loaded into the workspace. You may have to load trainrun.mat. hmm_train also animates the mean poses for each hmm and displays these as it runs.

Hmm_main will output logL's for the walking HMM and running HMM for the given test sequence.

Descriptions:
This is a 2-state HMM, with diagonal covariance for the 123 dimensional data. Expectation-Maximization learning was implemented using the JTA representation.

hmm_train.m - Input: none.
This program is responsible for calling the hmm. It sets up all the variables and preprocess variables from the 4 JTs. This program also saves the JT parameters generated by each HMM and saves them to files in the current MatLab Workspace. The A, pi and

eta matrices for the walking HMM are stored in VARW.MAT and the same matrices for the running HMM are stored in VARR.MAT. This way, the training doesn't have to be run every time I want to test a new sequence. When I test, I just load these two .MAT files and test using these parameters.

hmm.m - Input: (train, i) where train is the entire training cell array and i is the index (1-4) of the sequence in the cell array to be trained on
This program implements the EM algorithm across the JTs. It uses the diagonal covariance matrix. Th eE step involves collecting from the q0y0 node to the qT-1qT node right to left, and then distributing across all nodes. After distribute, the log likelihood (logL) is calculated by summing all values in the final backbone node, qT-1qT. If the logL is still above threshold, the iterations will continue. The M step is then performed, using the update rules provided in the lecture notes. 2 functions are included here in order to implement the numerical trick to avoid underflow and overflow.

hmm_main.m - Input: (test, i) where test is an entire cell array and i is the index (1-4) of the sequence to be tested
This program tests the desired sequence on both HMMs. It loads the aforementioned VARW.MAT and VARR.MAT files, which are the parameters for the walking HMM and running HMM respectively. This program calls hmm_test to run the test sequence on the JT for each HMM. Finally, it reports the logL's for both HMMs which can be compared. The larger logL is used to classify the test sequence as part of that specific class.

hmm_test.m - Input: (test, trans, priors, means)
test - a single sequence from the original test cell array (either running or walking)
trans - the A matrix, transition prob matrix generated by hmm_train
priors - the pi matrix, generated by hmm_train
means - the state by T dimension matrix, means for each coordinate across the time points, generated by hmm_train
This program runs the given test sequence through a JT defined by the 3 parameters. The code is pretty similar to that in hmm.m since I just perform collect and distribute again for each iteration. The logL reported is the likelihood of this given sequence belonging to this specific class as defined by A, pi and eta.

Notes & Results:

Unfortunately, I was having some numerical problems. It seems for testing, many more iterations are required in order to converge. This is around 25 or 29 or so, thus testing takes some time. However, it does converge close enough to be deemed a classification. Also, when displaying the mean poses for waling and running, it seems the feet are somewhat too close together. I could not fix this problem. However, the two poses are different enough to attain 100% classification accuracy. Please see figures and table below.

	WalkingHMM LogL	RunningHMM LogL	My Classification	TRUE
testw{1}	-5.35E+03	-3.98E+04	Walking	Walking
testw{2}	-5.35E+03	-3.98E+04	Walking	Walking
testw{3}	-5.33E+03	-3.17E+04	Walking	Walking
testw{4}	-5.26E+03	-3.96E+04	Walking	Walking
testr{1}	-1.87E+04	-6.62E+03	Running	Running
testr{2}	-1.83E+04	-8.22E+03	Running	Running
testr{3}	-1.97E+04	-7.34E+03	Running	Running
testr{4}	-1.60E+04	-6.25E+03	Running	Running

Another issue is in regards to the logL's. I wasn't sure if they should be further apart for a given sequence. Perhaps it is not the maximal distance between the two classes; nevertheless, 100% accuracy is obtained when choosing the larger logL as the specific class.

Another issue was related to the trainr sequence. MatLab kept complaining about a predefined trainr function. For this reason, I have included trainrun.mat which is just trainr renamed. I didn't have time to look into any of the details of this.

I have attached 2 images, the mean walk and the mean run generated by my hmm.m after training on all 4 sequences for walking and running respectively. One can tell by the distribution of points, that the runner is leaning forward, hands and elbows are raised together and the knees are slightly raised as well.

Aside from these problems, I am happy with the performance. It classifies with 100% accuracy and does so in decent time. If I had more time, I would make things run more smoothly.

BONUS:
In a recent ICAASP 2005 paper, Wang et al. provided methods for processing 3D motion capture data for dance movements. This problem is much more difficult since it requires following two people and intricate movements. As with most data sets, some form of smoothing would have to be used on the points, otherwise there is simply too much noise. Their parameters included the combined joint angle space; basically angles between various important body limbs, such as arm and torso. 22 angles were used to characterize an individual. In addition to these parameters, a shape descriptor was used, which describes the shape of the image with respect to each tracked point. In order to capture the spatial representation of this data, simple 2D tables are not sufficient. In this case, circular shells and sectors were used to describe the distribution of points around a different given point. The shape descriptor is thus represented by a histogram, calculating the scattering of points in 2D space. This model performed well, but it is still difficult to discretise the points across a multivariable domain while increasing

performance over a high frames per second rate. I didn’t have time to implement any of this.