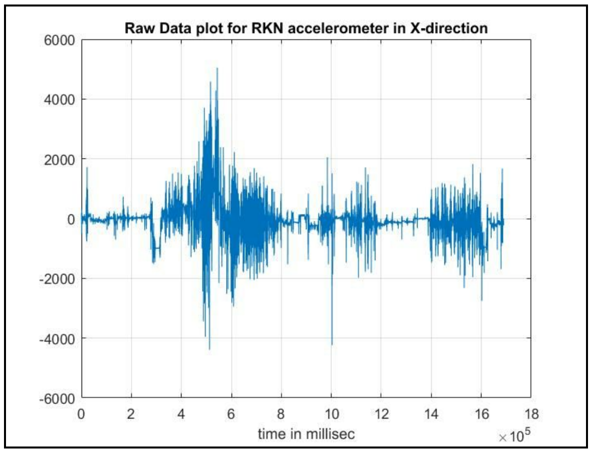
ACTIVITY RECOGNITION USING HMM

INTRODUCTION:

Activity Recognition is a complex and an important problem with many applications such as Human-Computer Interaction, Sports training, etc. Most of the existing models try to recognize only human activities of certain types [1]. That is, there is no general high level model for activity recognition and recognition. Several database are also available. However, most of them contain data specific to certain types of activities [4]. The lack of more general database and models for activity recognition can be explained by the complexity in conceiving and recording a dataset that accurately represents the day-to-day activities. We use the Opportunity Dataset - a dataset that comprises of complex naturalistic activities with an enormous number of atomic activities [4]. For this project, we use a subset of this dataset - columns 1 through 37 as input and column 245 as the high-level activity that is to be predicted. For the purpose of this project, we will be using Hidden Markov models to predict upper level activities based on sensor data.

DATASET DESCRIPTION:

The data has been collected from a sensor rich environment which comprises of large number of atomic activities. The 12 3-axis accelerometer sensors and 7 Inertial measurement Units are placed at different locations of the body. Opportunity dataset provides data for five subjects, each subject is allowed to have five ADL runs, which is to perform multiple high level tasks tasks inside the environment (such as *relaxing, coffee time, early morning, cleanup, sandwich time* and NULL) and the data has been collected. The Raw data of one of the sensor is plotted as shown below.



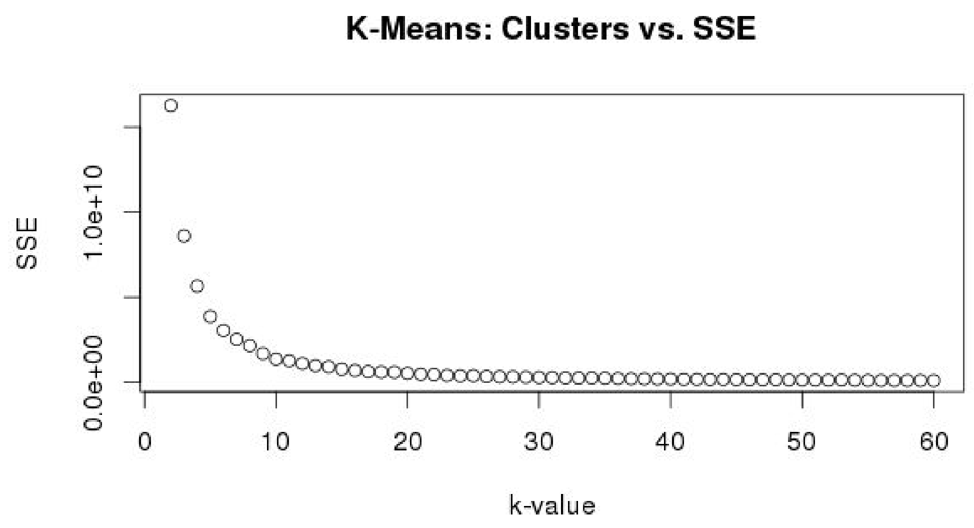
METHODOLOGY:

STEP 1: DATA PREPROCESSING:

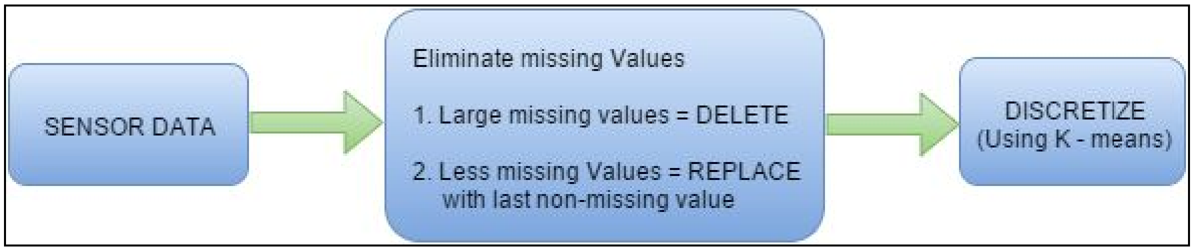
Training and Testing data:
Datasets ADL-1, ADL-2 and ADL-3 were used for training and cross-validation purposes, and ADL-4 and ADL-5 were used for testing purposes.

Method:
During preprocessing, we eliminate the columns 35, 36 and 37 since it contains mostly NULL values in all the 5 Datasets (over 95% of the data is missing). Columns 14, 15, and 16 are also eliminated because the datasets ADL-2 and ADL-5 have mostly NULLs for these columns. Not eliminating these sparsely populated columns or interpolating such columns would result in a highly biased model.

The NULLs in the other columns are interpolated using *LOCF (Last Observation Carried Forward) Interpolation method.* After the missing data values are interpolated, we use the sliding window of 500ms with a step size of 250ms and compute the mean of each sensor value in this window. Class is determined by the mode of all class values in the list. This reduces the overall size of the dataset drastically, while still preserving important information. Since dealing with continuous attributes yields an extremely complex model, with very low probability values, the next step is to cluster the data. We use K-Means clustering for this method. The following graph shows the Sum of Squared Error in the data for different cluster sizes:



Based on the elbow plot, we took 50 as our number of clusters, because further clustering adds unnecessary complexity without improving the accuracy, while lower number of clusters would lead to too much loss of information. Since the data is made up of overlapping clusters, it makes sense to set a high j value.

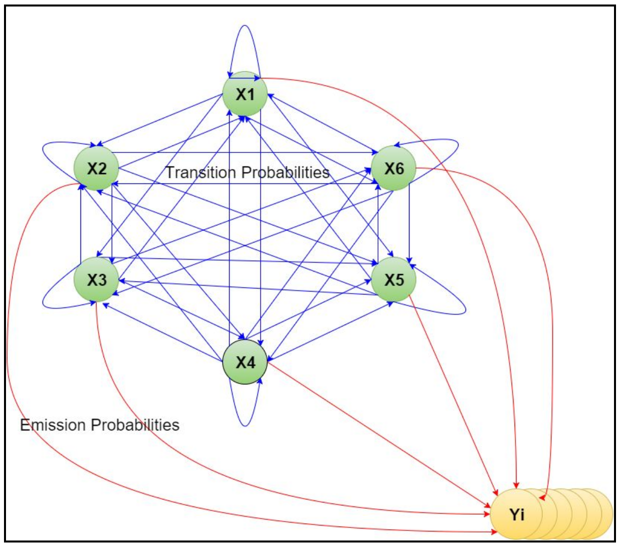

SENSOR DATA
Eliminate missing Values
1. Large missing values = DELETE
2. Less missing Values = REPLACE with last non-missing value
DISCRETIZE (Using K - means)

Step 2: TRAINING, CROSS VALIDATION

Hidden Markov Model

We use HMMs to train the model because of its importance and widespread use in modelling time series data. Other than Activity Recognition, HMMs are also widely used many areas of Artificial Intelligence such as Speech Processing, Pattern Recognition, Computer Vision, etc. [3]. A Hidden Markov Network gets its name from two key properties. First, it assumes that the observation at time t, the observation Y_t, was generated by some state X_t, that is hidden from the user. Second, it assumes that the given the state X_{t-1}, X_t is independent of all other non-descendants - that is, the Markov property. The output, Y_t, is also independent of all other states and previous observations given X_t.

The HMM for the Activity Recognition Problem is shown below. All the hidden states (X_i) are connected to each other, while all the observed values Y_t, or outputs are connected to all the hidden states. In our project, the number of hidden states is equal to the number of classes we need to classify, i.e, 6. The number of observed variables is equal to total number of possible observations, which is the clustered number of values, i.e., 50.



2.1 Training

For training, we performed two approaches. Initially. we trained the model by directly computing the each probability value similar to a Bayesian approach. Once the values of the three parameters (described below) are computed, we can use the Viterbi algorithm to find the Activity or Class with the highest probability for a given sequence of observations. In this case, each sensor is considered independent of each other. However, we noted that the model trained in this way is not achieving good prediction.

For the second approach, we used the Baum-Welch Algorithm, which is a dynamic programming based approach to estimate the value of the parameters, that is, π, A and B, where π refers to the initial state distribution probabilities. It is computed as shown below:

$$\pi_i = P(X_1 = i)$$

A refers to the state transition probabilities, in which each a_{ij} is the probability of going to state 'j' from state 'i', and is computed as shown below:

$$A = \{a_{ij}\} = P(X_t = j | X_{t-1} = i)$$

and B refers to the emission matrix, where each $b_j(y_t)$ refers to the probability of emitting O_j from state S_i:

$$b_j(y_t) = P(Y_t = y_t | X_t = j)$$

For the Baum-Welch algorithm, before the first iteration, the values of π, A and B are assigned random values, and the algorithm is repeatedly executed for sufficient number of iterations or until convergence. Each iteration of the Baum-Welch algorithm proceeds in the following manner:

Forward Step:

In the forward step, we compute the value of α which is given as:

$$\alpha_i(t) = P(Y_1 = y_1, ..., Y_t = y_t, X_t = i | \theta)$$

Backward Step:

In the backward step, we compute the value of β which is given as:

$$\beta_i(t) = P(Y_{t+1} = y_{t+1}, ..., Y_T = y_T | X_t = i, \theta)$$

To update the three parameters, the forward and backward step of each iteration is followed by the updation step, where we compute some temporary values needed to update the parameters as:

$$\xi_{ij}(t) = P(X_t = i, X_{t+1} = j | Y, \theta) = \frac{\alpha_i(t) a_{ij} \beta_j(t+1) b_j(y_{t+1})}{\sum_{k=1}^{N} \alpha_k(T)}$$

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i, j)$$

Now the three parameters can be updated as:

$$\pi_i^* = \gamma_i(1)$$

$$a_{ij}^* = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)}$$

$$b_i^*(v_k) = \frac{\sum_{t=1}^{T} 1_{y_t = v_k} \gamma_i(t)}{\sum_{t=1}^{T} \gamma_i(t)}$$

This procedure is repeated number of times till either a convergence is reached, or a large number of iterations are run. The updated parameters, (π, A and B) are then used for validation and testing.

2.2 Cross Validation:

Three fold cross validation was performed, using the training procedure described above. The entire training dataset was randomly sampled (without replacement) into 3 mutually exclusive subsets and training was performed thrice, once on each of these pairs of subsets. In our method, these results are validated by the third subset, and then the set of parameters that give the best accuracy are chosen as the optimal parameters. Another possible approach is to average the outcomes of each training step in the cross validation. Stratified random sampling was also attempted with the training data, but we did not observe any appreciable improvement in performance.

STEP 3: TESTING

Viterbi Algorithm:

After calculating π, A and B from Baum Welch algorithm and performing cross-validation as described above we can use the Viterbi algorithm to compute the most probable state sequence. Viterbi algorithm is based on $V_{t,k}$, is the probability of most probable state sequence that is responsible for t=1 to t = t observations, with k as its final state, and is computed as shown below:

$$\begin{aligned} V_{1,k} &= \mathrm{P}\big(y_1 \mid k\big) \cdot \pi_k \\ V_{t,k} &= \max_{x \in S} \big(\mathrm{P}\big(y_t \mid k\big) \cdot a_{x,k} \cdot V_{t-1,x}\big) \end{aligned}$$

In Viterbi algorithm, at each timestamp, the most probable state, and its probability value that can be achieved from the previous timestamp are calculated. Once all the timestamps are completed, then we define the path to be corresponding to the state with maximum $V_{t,k}$ value at the final time stamp. Given the parameters of the data (π, A, B), the Viterbi algorithm is a dynamic programming algorithm that estimates the most likely sequence of hidden states (known as the Viterbi path) that can arise from the sequence of observed events. Based on the (π, A, B) values we obtained from the training step, Viterbi algorithm is implemented to find the best sequence of possible states.

RESULTS

After performing training on the data using the two methods specified before, we observed that the accuracy for the simple bayesian approach, which involves computing each of the probabilities based on the counts of the variables, was very low, (less than 10%) for the test dataset, even after cross validation. After this, we implemented the Baum-welch algorithm, with random initial probabilities to train the initial parameters. However, we observed that very low probability values, caused due to normalization and increasing number of training samples, and the fact that Baum Welch involves multiplication of probabilities, means that these probabilities will soon become too small for machines to compute. As a result, we computed the logarithmic probabilities, as described by Mann in [8]. This version of the model gave slightly better results, ranging between (20- 36%) accuracy, based on random initialization, and random subsetting during 3-fold cross-validation. The accuracy of the model is computed by:

$$overall\ Accuracy = \frac{\{Number\ of\ Matching\ Samples\}}{Total\ number\ of\ samples}$$

Weighted F-measure is computed by computing precision and recall for each state. The formula for weighted F-measure is given by,

$$Weighted\ F-Measure = \sum_{all\ classes} \frac{precision-of-that-class * recall-of-that-class}{precision-of-that-class + recall-of-that-class}$$

and the corresponding precision and recall are given by the formulae:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \text{ and } Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

While we achieved very low accuracies (between 11-20%) without cross-validating the results, upon performing 3-fold cross validation, we achieved better probability.
The accuracy of each of the folds when performing training with the other pair, is as shown below:

3-Fold Cross Validation Accuracy:

Fold1 Accuracy	**Fold2 Accuracy**	**Fold3 Accuracy**
31.54%	12.97%	19.85%

By choosing the first fold parameters as our best parameters, when prediction was performed on the test data, the accuracy achieved was 36.05%, which is slightly above the baseline value (which is computed as the percentage of majority class, which in the test case of ADL4 + ADL5 is 36.03%).

Based on computing the precision, and recall for each class, based on the formulae specified above, the weighted F-measure computed has the value 0.2638015

ADDITIONAL WORK

In addition to cross-validating our training data, we also performed a comparison in terms of accuracy with other models such as KNN classifier and Naive Bayes classifier. The **Naive Bayes classifier** gave us an accuracy of nearly **36%,** but it was merely classifying all activities as ‘Sandwich Time’, which is 36% of the test dataset. We also used the recursive partitioning decision tree (Rpart), but owing to the size of the data, and number of objects, the package we used in R kept crashing, and didn’t complete execution.

ANALYSIS OF RESULTS

As it can be seen, in all the different methods we have tried, the accuracy results are not very high (equal to or slightly greater than baseline). There are several factors we need to take into consideration in this data:

1. Accuracy of the Baum Welch algorithm is depending upon the initialization parameters - traditional desktops don’t have the computational power to run this for large number of iterations for convergence.
2. Accuracy of the result depends heavily upon the choice of discretization measure, and the number of discrete values entire dataset is converted into. Since we are using K-Means in this method, which performs poorly with respect to overlapping, different shaped and noisy clusters, how well K-Means performs is heavily dependent upon the quality of the data.
3. Accuracy of the result depends upon the quality of the interpolation measure.
4. Usage of log probabilities also reduces the accuracy of the results, as suggested by Mann [8].

CONCLUSION

In conclusion, we were able to build a Hidden Markov Model, which can predict the high level labels directly from the sensor data, without actually using the intermediate atomic labels. While we haven’t achieved high accuracy in the results, we attribute this primarily to the usage of log probabilities based on Mann’s method [8], which, is an approximation measure, and we lose some precision, which will go a long way in computing the emission probabilities. Increasing the number of iterations of the algorithm helped in improving the accuracy, but increasing it beyond a certain point was not feasible given the limited computational resources. The k-fold cross validation method we performed was based on random sampling without replacement. However, stratified random sampling might be a better approach.

REFERENCES:

[1] de León, Rocío Díaz, and Luis Enrique Sucar. "A graphical model for human activity recognition." In *Progress in Pattern Recognition, Image Analysis and Applications*, pp. 350-357. Springer Berlin Heidelberg, 2004.

[2] Hamid, Raffay, Yan Huang, and Irfan Essa. "Argmode-activity recognition using graphical models." In *Computer Vision and Pattern Recognition Workshop, 2003. CVPRW'03. Conference on*, vol. 4, pp. 38-38. IEEE, 2003.

[3] Ghahramani, Zoubin. "An introduction to hidden Markov models and Bayesian networks." *International Journal of Pattern Recognition and Artificial Intelligence*15, no. 01 (2001): 9-42.

[4] Chavarriaga, Ricardo, Hesam Sagha, Alberto Calatroni, Sundara Tejaswi Digumarti, Gerhard Tröster, José del R. Millán, and Daniel Roggen. "The Opportunity challenge: A benchmark database for on-body sensor-based activity recognition." *Pattern Recognition Letters* 34, no. 15 (2013): 2033-2042.

[5] Cao, Hong, Minh Nhut Nguyen, Clifton Phua, Shonali Krishnaswamy, and Xiaoli Li. "An integrated framework for human activity classification." In *UbiComp*, pp. 331-340. 2012.

[6] Cilla, Rodrigo, Miguel A. Patricio, Jesús García, Antonio Berlanga, and Jose M. Molina. "Recognizing human activities from sensors using hidden markov models constructed by feature selection techniques." *Algorithms* 2, no. 1 (2009): 282-300.

[7] Nair, Naveen, Ganesh Ramakrishnan, and Shonali Krishnaswamy. "Enhancing activity recognition in smart homes using feature induction." In *Data Warehousing and Knowledge Discovery*, pp. 406-418. Springer Berlin Heidelberg, 2011.

[8] Mann, Tobias P. "Numerically stable hidden Markov model implementation." *An HMM scaling tutorial* (2006): 1-8.

[9] Li, Xiaolin, Marc Parizeau, and Réjean Plamondon. "Training hidden markov models with multiple observations-a combinatorial method." *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 22, no. 4 (2000): 371-377.

[10] Opportunity Dataset: Available at https://archive.ics.uci.edu/ml/machine-learning-databases/00226/

[11] Image source: Wikipedia articles on Baum - Welch and Viterbi Algorithms