Sensor-based Semantic-level Human Activity Recognition using Temporal Classification

Weixuan Gao gaow@stanford.edu
Chuanwei Ruan chuanwei@stanford.edu
Rui Xu ray1993@stanford.edu

I. INTRODUCTION

Human activity recognition has always been an active research topic driven by the potential ranging from health care, assistive technologies to manufacturing and smart surveillance system. In this project, we build several models to estimate the semantic-level human activities given multi-variate time series data from body-worn sensor. We use the K-Nearest Neighbors, Boosted Tree and Random Forest to build the baseline model, which transform the raw sensor data into the semantical labels without considering temporal correlation. Random Forest gives an accuracy of 86.1%, which is the highest among these methods. To include the temporal correlations, we firstly try to deploy the LSTM model directly on raw sensor data. Given the high variance and high dimensionality of raw sensor data, the LSTM fails to converge. In order to solve this problem, we build up a two-stage framework. The first stage is to estimate the low-level actions, which are provided by original dataset, from raw sensor data by traditional methods. There are three kinds of low-level actions: locomotion, left arm action and right arm action. For each multi-classification tasks, the accuracy are 94.1%, 82.5% and 74.4% respectively. The second step is to use the temporal sequences of low-level actions to estimate their corresponding semantic labels. Considering a combination of the temporal sequence of low-level actions as a sentence, the LSTM model has 75% accuracy. If output is a sequence of high-level labels, the LSTM model has 69.59% accuracy. However, without involving temporal relationships, the methods that directly project low-level actions to semantic labels have accuracy lower than 50%. Combining these two stages, the model has 68.2% accuracy.

II. RELATED WORK

Several works have been done to recognize simple actions from sensor data. For example, Ravi et al. [2005] used shallow classifiers, such as SVM, Nave Bayes, and SVM to classify eight daily actions, which achieved the accuracy of 95% in the result. Another work (Wang et al. 2012) implemented a Nave Bayes model to recognize six daily actions to offer music recommendation. However, because human activities are naturally complex, each single action is not isolated, but a sequence of actions in diverse combination. Therefore, capturing the temporal relations among actions is the key to solving the problem. Although much effort has been put in similar work on video frame recognition, it is extremely hard to directly transfer video-based method to sensor-based data due to the large difference on data format and the way of interpretation [3].

III. DATA SET

A. Data Introduction

The dataset OPPORTUNITY Activity Recognition Data Set is obtained from UCI machine learning repository. The dataset comprises the readings of motion sensors recorded while users executed typical daily activities. The dataset is recorded in a sensor-rich environment which contributes to 242 attributes. More than 27,000 of atomic activities were collected in each environment. In addition, a natural execution of activities is achieved by instructing users to follow a high-level script but leaving them free interpretation as for how to achieve the high-level goals. The dataset is designed to research work on the different level of activities: from simple gestures to high-level compound activities (see Figure 1). This hierarchy is able to be reflected in this data set.

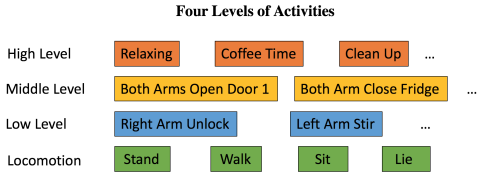

Fig. 1. Four Levels of Activities

B. Data Process

1. Kernel Smoothing Denoising is necessary for this research, because of the quality of equipment, and tiny and redundancy features. We use kernel smoothing method to filter noise and reduce the influence from unrelated information. The so-called Nadaraya-Watson kernel-weighted average is used,

$$\hat{f}(x_0) = \frac{\sum_{i=1}^{N} K_\lambda(x_0, x_i) y_i}{\sum_{i=1}^{N} K_\lambda(x_0, x_i)}$$

with the *Epanechnikov* quadratic kernel

$$K_\lambda(x_0, x) = D\left(\frac{|x - x_0|}{\lambda}\right),$$

with

$$D(t) = \begin{cases} \frac{3}{4}(1-t^2) & |t| \leq 1; \\ 0 & otherwise. \end{cases}$$

The size of the 40-nearest-neighbor smoothing window adapts to the local density of the x_i.

C. Data Scaling

In general, we observe that there are always differences between two subjects when doing the same gesture. Different people with different physical conditions, namely height and powder, influence the value of each sensor a lot. Thus, it is important to normalize each feature. We scale all the feature to be between 0 and 1.

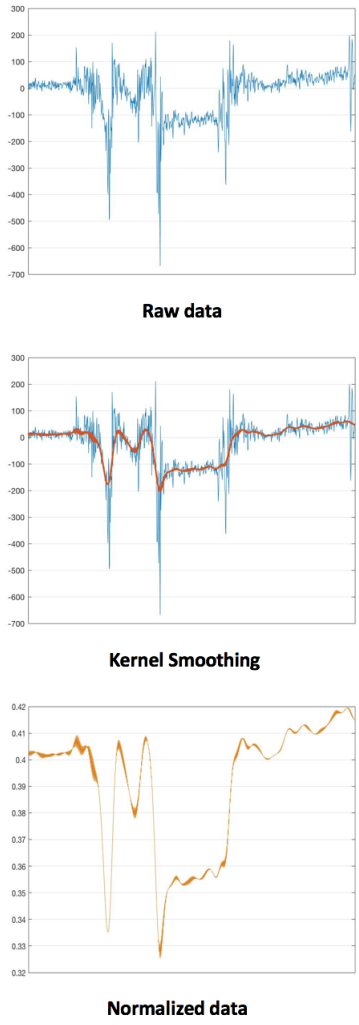

Fig. 2. Data Denoising and Scaling

IV. Method

A. Model Evaluation

The high-level activities has 5 non-null classes and 1 null class. The evaluation metrics we chose are accuracy and weighted F-score. The accuracy is defined as the proportion of correct prediction in all prediction. But the accuracy is limited in evaluating the classification result. The other metrics such as precision, recall, F-score and AUC will give more useful information. Since our problem is of multi-classification, as suggested by Chacarriage, R. et al., we use the weighted F-score to evaluate our model as it could consider take account the precision, recall and the imbalance of labels. The $precision_i$ is defined as $TP/(TP+FP)$, and the $recall_i$ is defined as $TP/(TP+FN)$. Thus the weighted F-score could be thought as the weighted harmonic mean of precision and recall, while the w_i is the proportion of samples of class i.

B. Baseline

The data in its nature are time-series data, however, because of complexity of sensor data we decided to start with methods which assume independence of training data. Thus, the models receive raw sensor data at each time step and predict the semantic labels. Although this assumption is seemly too contrived, it turns out they are still able to capture some key structures of the data. As shown in the

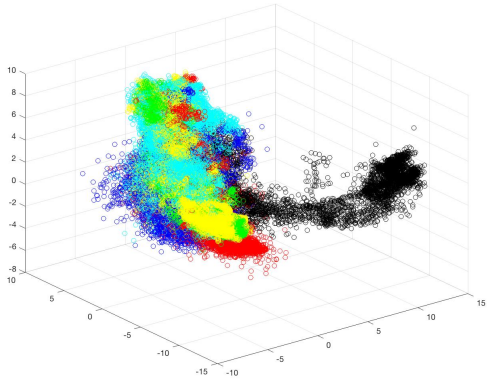

Fig. 3. PCA on Raw Data, Each Color Represents a High-level Label

visualization using PCA, the raw data is obviously not linear separable, we focus on the following non-linear supervised learning algorithms.

1) KNN: Given a new data, K-nearest neighbor methods finds the k nearest training data points according to a distance metric and predict the response by taking average.

$$\hat{Y}(x) = \frac{1}{k} \sum_{x_i \in N_k(x)} y_i,$$

where N_k is the neighborhood of x defined by the k closest points x_i in the training sample. In our experiments, we select the euclidean distance. And the output classification is chosen to be the majority class of nearest neighbors.

2) Boosting Trees: Given the high variability and non-linearity of sensor data, the regression tree should be an ideal method to try. As a tree-based method, the regression tree partitions the feature space into many small rectangles. And in each rectangle we could fit a simple model. After

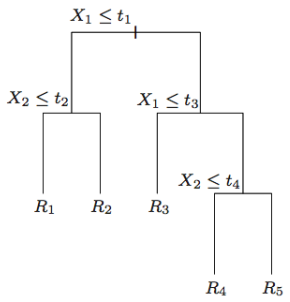

Fig. 4. Decision Tree

partitioning the tree, we could use the learned partition rule to predict.

The model is of the form:

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m).$$

The R_m is the mth region, c_m is the predicted response in that region. However, since finding the optimal splitting criteria is known to be NP hard, fitting the model requires greedy algorithm which firstly fits a large tree with optimal split at each level separately then use node impurity to prune the large tree. The most popular node impurity measure used are of the following form:

- Misclassification error:

$$\frac{1}{N_m} \sum_{i \in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk}(m).$$

- Gini index:

$$\sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_m k).$$

- Cross-entropy or deviance:

$$-\sum_{k=1}^{K} \hat{p}_{mk} log \hat{p}_{mk}.$$

Although regression tree is simple yet powerful, it still needs some improvement to tackle complex prediction problem. Boosting tree utilizes the advantages of regression tree and gives it more predictive power. Boosting tree repeatedly grow many shallow trees to the residuals and ensemble the trees in an additive fashion. We use the gbm package in R to implement the boosting tree algorithm.

3) Random Forest: Similar to Boosting Tree, the Random Forest is an ensemble method which needs to fit many trees. Different from boosting tree, the random forest trains many deep trees instead of shallow trees, and the finally reports the average of predictions. The bootstrapping helps reduce the variance of the estimators, while the random selections of features helps reduce the correlation of each trees. Thus the Random Forest on the one hand could learn complex structures of data which reduce the bias, on the other hand could alleviate the high variance from which deep trees suffer. Given the complexity and variance of our data, we decide to use the Random Forest. The experiments were done in Matlab using Statistics and Machine Learning toolbox.

C. Recurrent Neural Network

To include the temporal relationships in the model, we use the Recurrent Neural Network (RNN). Theoretically, the RNN is feasible to consider all previous actions in the training sequence which is particularly suitable for our purpose because each high-level activity is usually about 1000 time step(about 30s). And it might not be very appropriate to cut the sequence into small pieces.

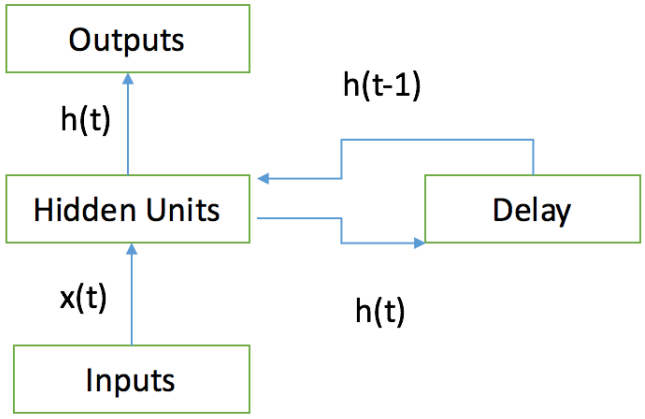

Fig. 5. A Simple Fully Connected RNN

But the simple RNN might not actually be able to learn the long dependence of the training data. We thus apply the Long Short-Term Memory(LSTM) layer in our model.

Compared with simple RNN, the LSTM has some additional structure. It has input gate, forget gate, output gate, new memory cell and final memory cell. This complex structure gives LSTM the ability to learn the long dependency and makes it preferable in our applications.

We firstly try to use the preprocessed raw sensor data directly. But this method fails to converge given our limited computing resources. We decide to simplify the model by inputting the sequence of low-level actions instead of raw sensor data. We consider three kinds of low-level activities: locomotion, left arm actions and right arm actions. We encode each unique combinations of those actions as a category. Thus the whole network is consisted with three layers: an embedding layer, a LSTM layer and a softmax layer.

To improve the RNN model, we also add a convolutional layer with max-pooling after the embedding layer because we guess the actions in a small time step range should be highly correlated while convolutional layer might be able to capture this information.

This model is still limited since it could only give one high-level label after observing the whole sequence. Since we actually want to have a prediction of high-level activities from the beginning of the sequence, we move forward by

TABLE I

ACCURACY OF EACH MODEL ON SUBJECT 1

	K-NN	Boosting Tree	Random Forest
Raw to locomotion	86.59%	92.17%	94.1%
Raw to left are low-level activity	82.66%	84.78%	82.5%
Raw to right arm low-level activity	72.15%	71.73%	74.47%
Raw to high level activity	76.4%	79.72%	86.01%
low level to high level activity	46.7%	44.7%	45%

TABLE II

RANDOM FOREST MODEL ON SUBJECT 1,2, AND 3

	Accuracy	F1 score
Raw to locomotion	89.05%	0.8889
raw to left arm low-level activity	79.59%	0.8258
raw to right arm low-level activity	76.93%	0.7813
raw to high level activity (with null)	82.68%	0.8286
raw to high level activity (no null)	84.09%	0.8442

considering outputting the sequence of high-level labels. This is easily established since the design of RNN naturally supports such output.

All the RNNs are implemented using the Deep Learning library Keras in Python, and we use the Tensor Flow as its backend.

V. RESULT AND ANALYSIS

A. Baseline

Considering the long training time with full data sets, we ran the experiments on subject 1 first. The adl1, adl2, adl3 and adl4 are training set and the adl5 is the test set.

According to the results, we decide to use Random Forest on the full data sets. The training data are from subject 1, 2 and 3. Their adl5 are used as test data, while other adl are used as training data.

Surprisingly, we find that the increase of sample size does not increase the accuracy but indeed decreases it. We guess it is because of the variance between different subjects.

The test accuracy gained from LSTM model using low-level actions as input is not very satisfactory but it proves our guess that it is possible to use sequences of atomic actions to predict the semantic information.

TABLE III

MODEL EVALUATION OF EACH RNN ON SUBJECT1,2 AND 3(WITHOUT NULL)

		Training Accuracy /F1-score	Test Accuracy /F1-score	Test Accuracy (with estimated low-level)/ F1-score
Many to One	LSTM	72.13%/0.6526	50%/0.4545	68.75% /0.6087
	CNN+ LSTM	77.52%/0.7586	75.5%/0.7500	68.5%/0.6875
Many to Many	LSTM	81.07% / 0.8081	69.59%/0.7027	74.6%/0.7541
	CNN+ LSTM	73.23%/0.7163	62.96%/0.6439	61%/0.6101

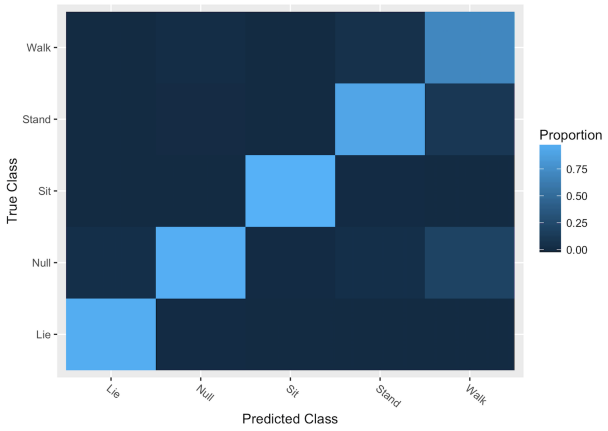

Fig. 6. The Confusion Plot for Locomotion on Subject 1, 2, and 3

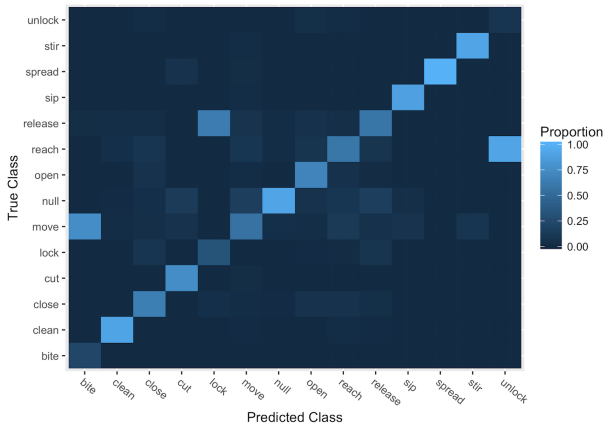

Fig. 7. The Confusion Plot for Right Arm Low-level Activities on Subject 1,2, and 3

It should be noted that the result of many to one models should not be directly compared with the baseline. In this model, we only output one label given a sequence while the baseline model output labels at every timesteps. So this many to one models here only serves as a proof that the temporal relations in low-level actions could indeed be used to predict semantic level activities. Also, since each high-level activity is about 1000 time steps, by using the one label per sequence, we only have 16 test sequences. Thus the results are not very stable.

After testing our initial guess, we work on predict the high-level labels at each time step using many to many framework. The many to many models have slightly better test accuracy. We think it might because the increased outputs frequency give could give more information of error in training despite the fact that the initial predictions of a sequence could hardly be accurate(no enough temporal information to use at the beginning).

Also, one interesting point is that using estimated low-level activities do not significantly impair the test accuracy

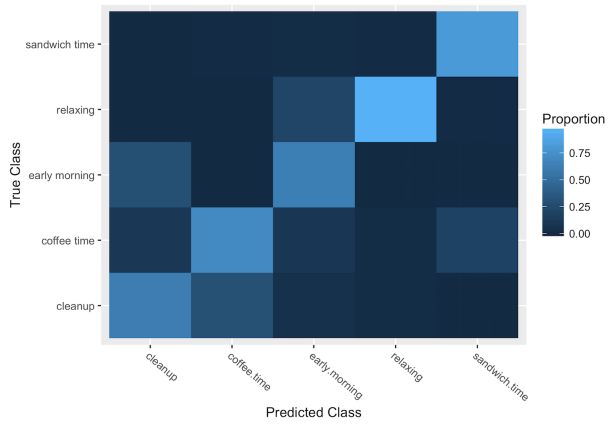

Fig. 8. The Confusion Plot for High-level Activities Using LSTM (many to many)

despite that the estimated low-level activities itself are not of very high accuracy. But it is not impossible as the subjects in the data sets were only assigned to perform certain activities without rigid scripts. Thus the temporal relationships from low-level actions to high-level activities are of variability. This result also suggests that to obtain more accurate results, we need to include more information beside the low-level activities.

VI. Conclusion and Future works

In this project, we study and apply different machine learning techniques to solve multivariate classification problem by seeking temporal relatedness veiled in the dataset. Although the outcome of baseline model surprisingly outperformed two-stage RNN model, we still remain optimistic that the latter has potential to give out better result under more completed modeling. In the future ,we are going to focus on improving the current model by providing more sophisticated modeling and extracting veiled temporal relatedness from either a single attribute or interaction of several different attributes of low-level activities, which needs a new form of representation such as adjacency-matrix in graph theory. In addition, more data are needed to feed the neural network. More progressively, we would like to build a system which could directly learn the temporal-relatedness from raw data and generate the predictions of semantic levels. Thus we do not need to label the other low-level actions, which is not very practical if we want to have very large data sets to train our model.

References

[1] Ye. L, Liaing N, Lei. H, Luming. Z, and David. R, (2016). Action2Activity: recognizing complex activities from sensor data, IJCAI'15 Proceedings of the 24th International Conference on Artificial Intelligence pp 1617-1623.

[2] Nishkam. R, Nikhil. D, P. Mysore, and M. L. Littman, Activity recognition from accelerometer data, In Proceedings of the Conference on Innovative Applications of Artificial Intelligence 2005.

[3] Kuehne. H, Gall. J, & Serre. T, (2016). An end-to-end generative framework for video segmentation and recognition,2016 IEEE Winter Conference on Applications of Computer Vision (WACV).

[4] Breiman. L, Random Forests. Machine Learning,45, pp. 532, 2001.

[5] Hastie. T, Friedman. J, & Tibshirani. R, The Elements of Statistical Learning.Springer Series in Statistics, 2001.

[6] Sepp.H,& Jrgen. S, Long Short-Term Memory, Neural Computation, Volume: 9, Issue: 8, Nov. 15 1997 .