**Machine Learning Classification for Human Activity Recognition**

Andy Malinsky, Dheemanth Rajakumar, Greg Moore

Shiley-Marcos School of Engineering, University of San Diego

AAI-501: Introduction to Artificial Intelligence

Professor Andrew Van Benschoten, PhD.

Professor David Friesen, MS

08/12/2024

**Machine Learning Classification for Human Activity Recognition**

Human Activity Recognition (HAR)

Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text

**Introduction and Dataset**

This report reviews and compares three different machine learning models using the Human Activity Recognition Trondheim (HARTH) dataset from UC Irvine which is a culmination of three-axial accelerometer sensor data on 22 test subjects (people). The sensor data was collected from two different sensors, one on the thigh and one on the hip, from people doing normal things during a 2 hour period of time in their homes. The data has been annotated to assist in the development of machine learning classifier models for processing timestamped HAR sensor data. From the 22 subject files combined, there are 6461328 rows of sensor data that the three models developed for this final project were used to train and evaluate the machine learning models. The three models developed to process the HARTH data are a Convolutional Neural Network (CNN) model, a Support Vector Machine (SVM) model, and Bidirectional Long Short-Term Memory (LSTM) model. Each of the 22 subject test files in the dataset contains 8 features or different sensor data that is timestamped at 50Hz to 100Hz and labeled. This data was used to train the models and test/validate the model performance. The features in each dataset .csv file are:

timestamp: date and time of recorded sample

back\_x: acceleration of back sensor in x-direction (down)

back\_y: acceleration of back sensor in y-direction (left)

back\_z: acceleration of back sensor in z-direction (forward)

thigh\_x: acceleration of thigh sensor in x-direction (down)

thigh\_y: acceleration of thigh sensor in y-direction (right)

thigh\_z: acceleration of thigh sensor in z-direction (backward)

label: annotated activity code

The annotated activities in the dataset are:

1: Walking

2: Running

3: Shuffling

4: Stairs (Ascending)

5: Stairs (Descending)

6: Standing

7: Sitting

8: Lying

13: Cycling (sit)

14: Cycling (stand)

130: Cycling (sit, inactive)

140: Cycling (stand, inactive)

**Exploratory Data Analysis**

Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text

**Model Selection**

Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text

**Model Analysis**

Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text

**Support Vector Machine Model**

We implement a Support Vector Machine (SVM) model on the HARTH dataset. An SVM is an algorithm that creates hyperplanes, or decision boundaries, with the largest possible distance to sample points (Cortes & Vapnik, 1995). It is useful for supervised learning tasks, such as classification, which lends itself well to this dataset. The SVM model chosen for this project is implemented using the SGDClassifier estimator from the sklearn Python library (Buitinck et al., 2013). By default, the SGDClassifier fits a linear SVM with stochastic gradient descent learning. This means the gradient of the loss is estimated at each sample and updated based on the provided learning rate parameter. Based on the library documentation, this implementation works well with floating point values for the features, and the features of the HARTH dataset are all floating-point values, based on accelerometer sensor data.

For data preparation for the SVM model, the 22 datasets were combined into one, which totaled 6,461,328 rows. The label column values were mapped from integers to strings that included the integer label value as well as the corresponding activity text associated with that label value. We then split the data into independent and dependent features. The independent features include the six features related to the activity accelerometer sensor data: back\_x, back\_y, back\_z, thigh\_x, thigh\_y, thigh\_z. The dependent feature is the activity label. Next, for better results on the SVM model we apply normalization to the independent features. This scales the values of the independent features to fall within the range of 0 to 1. Exploratory data analysis of the labels depict that most labels are Sitting, Walking, or Standing. This makes sense as those activities are most common in daily life. The data was split into a training and test set, with a test size of 20%. The number of samples in the training and test set was 5,169,062 and 1,292,266, respectively.

We first run the model using the default parameters to get a baseline performance score. Default parameters of note used by the model include the loss function known as ‘hinge’ which gives a linear SVM, penalty is given a value of ‘l2’ which is the standard regularization term for linear SVMs, an alpha of ‘0.001’ which is the constant that multiplies with the regularization term and is used to compute the learning rate, the learning\_rate which is set to ‘optimal’ where the change in rate is determined by a preset heuristic, max\_iter value of 1000 which is the number of epochs over the training data, and early\_stopping which is set to False which is whether or not to terminate training when validation score is not improving. We also set a default random\_state value to support reproducibility and the n\_jobs parameter to -1 for faster compute time. We fit the baseline model on the training set, then ran predictions through the test set and comparing the predicted labels with the actual labels. The baseline model scored an accuracy score of 0.61, with a 5-fold cross validation resulting in an average baseline accuracy score of 0.59. The model had near zero values for precision, recall, and f1 for less prevalent activities such as Cycling (Stand), Cycling (Sit, Inactive), Cycling (Stand, Inactive), Stairs (Ascending), and Stairs (Descending), which shows that there needs to be a more even distribution of activity labels for a better performing model. To increase performance, we utilize a hyperparameter tuning by performing a grid search using 3-fold cross validation. We selected 3 alpha values of 1e-1, 1e-4, and 1e-7 to compare. We also tested the ‘hinge’ loss vs ‘log’ loss. The log loss function gives logistic regression, a probabilistic classifier loss function. We also tested early\_stopping. The best parameters that were found from the grid search were alpha: 1e-7, loss: log, and early\_stopping: False. The model with these updated parameters predicted on the test set with an accuracy of 0.66, with a 5-fold cross validation resulting in an average accuracy score of 0.64, which was an increase of 5% from the baseline model.

For our calorie experiment, we run the trained SVM model on each of the 22 separated datasets to predict the labels. Each activity label corresponds to a different metabolic calorie burn value which is calculated based on the metabolic equivalent formula for total calories burned. Taking the predicted labels from the model, Figure 1 shows the cosine similarity between the predicted and actual labels vs the variance of the actual labels, Figure 2 shows the calculated cosine similarity vs the absolute error between the burned calorie count from the predicted and actual activity labels, and Figure 3 shows the absolute error between the burned calorie count from the predicted and actual activity labels vs the variance of the actual labels.

**Figure 1**

*Scatterplot of cosine similarity vs label variance*

A graph with blue dots

Description automatically generated

**Figure 2**

*Scatterplot of cosine similarity vs total error*

A graph with blue dots

Description automatically generated

**Figure 3**

*Scatterplot of label variance vs total error*

A line graph with dots

Description automatically generated

Figure 1 displays that there is a strong negative correlation between label variance and cosine similarity. A lower variance in the activity labels results in a higher cosine similarity between predicted and actual labels, while a higher variance in the activity labels results in a lower cosine similarity. Figure 2 displays a negative relationship between total absolute error in calorie burn count vs cosine similarity. As cosine similarity increases, or the difference between the predicted and actual labels decreases, the total error decreases. Figure 3 demonstrates a positive relationship between label variance and total error. As label variance increases, total error increases. These results suggest that the model was able to predict better on datasets with fewer distinct activity labels. For example, dataset ‘012’ contained 311,722 samples of Sitting with a model prediction accuracy of 0.97, while dataset ‘029’ contained 121,460 Walking samples with only 3,406 sitting samples and had a model prediction score of only 0.3. These results show that the model is most likely suffering from overfitting with a bias toward most of the labels which is Sitting. This supports the finding that there were not enough training examples across all activities, and that the activities with the most labels are easier for the model to predict. Potential future directions to increase performance could in include utilizing bootstrapping or data augmentation to strive for a more even distribution of activity labels.

**Convolutional Neural Network (CNN) Model**

Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text Add Text

**Bidirectional Long Short-Term Memory (LSTM) Model**

Bidirectional Long Short-Term Memory (LSTM) is a Recurrent Neural Network (RNN) that keeps previous short-term memory and uses it for long-term training. There are three primary gates as part of an LSTM structure which include input, output, and forget type gates. The bidirectional LSTM can look at data forward and backwards to learn data from the future and past which makes it a more complex algorithm. The bidirectional LSTM is built by first defining a sequential model having a stack of layers where each layer has an input and an output. The first bidirectional layer gives the LSTM the ability to process data in forward and backwards directions finding patterns from past and future. The LSTM in the model uses 64 units that can sequence predict by learning long term patterns. A parameter was included in building the model to ensure that the input and output data are of the same length which is required for stacked LSTM layers that need full sequences as input. A dropout layer was added to prevent overfitting along with another bidirectional LSTM layer and another dropout layer. A fully connected dense layer was added as the final layer of the classification model to map the output to a probability. Lastly, the model was compiled, trained, and validated. The bidirectional LSTM is used for sequenced data analyzing the data in both directions. The K-Fold Cross Validation method was used for this model which splits the data into 5 different folds.

**Figure 1**

*Bidrectional LSTM Model Summary*

A screenshot of a computer

Description automatically generated

This bidirectional LSTM model was used to analyze the HARTH dataset comprised of 22 different sets of human activity accelerometer sensor data. The bi-LSTM model was trained and validated on the data and tested using a combined portion of known labeled HAR data. Several versions of the models were compiled, trained, and validated, but the Average Accuracy was typically around 87 to 89 as shown in Figure1. The results of testing predictions on some of the known data had mixed results as seen below. There was some good prediction performance seen on some of the activity labels such as standing, lying down, and cycling while seated as seen in Figure 2. But there were false predictions on these activities as shown in Figure 3.

**Figure 2**

*Bidrectional LSTM Model Correct Prediction for 6 - Standing*

A screen shot of a computer

Description automatically generated

**Figure 3**

*Bidrectional LSTM Model Summary False Prediction for 13 – Cycling (Sit)*

A close up of a computer screen

Description automatically generated

There also seemed to be different results using smaller and larger windows of sensor data for testing the model. There are areas that should be explored more to improve the model through model tuning parameters and understanding the data window sizes and how they affect the performance of the bidirectional LSTM model.

**THE METALYZER**

The application we chose to use our model for is an idea for a product named “THE METALYZER” that involves a hip or back based accelerometer sensor that attaches to a person. The METALYZER will record and process sensor data real-time and by implementing our classification model into the sensor device, it will predict what activity the person is doing. When the person moves to another activity, the METALYZER will process the previous activity data using timestamps, MET table, and weight (kg) and will convert the time duration of the activity to calories burned using a Metabolic EquivalenT translation table for the activities. One MET is equal to one minute of the amount of energy (calories) the human body burns when at rest, so we chose this standardized translation table for our sensor product. The formula for this calculation is:

Burned Calories = MET level of activity \* 3.5 \* Weight(kg) \* minutes of activity / 200

The METALYZER will start accumulation like a stopwatch recording and will store previous sessions in non-volatile memory for historical analysis and processing. The base Python code for this type of sensor product has been designed and lightly tested using some of the HARTH sensor data. The METALYZER can be used by anyone, but it could be very useful in the managed healthcare of elderly people in retirement homes. These devices could also be RF connected to a control station at the retirement home for monitoring. For example, if the METALYZER determines that a person is lying down in the daytime, a wellness check on the person can be made to ensure they have not fallen. If there is a classification for a person not lying down during sleeping hours, then a wellness check can also be done. This type of sensor device will greatly help in the management of healthcare and safety for individuals. The METALYZER will calorie-process the timestamped sensor data streaming to our classification model to determine the recorded activity until another activity is determined. The METALYZER can run on a small, replaceable battery and will sport a small display with a few buttons with indicators for low battery to ensure continuous use. The METALYZER can be used not only for classifying human activity and calorie counting, but also for safety purposes and healthcare.

**References**

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., Vanderplas, J., Joly, A., Holt, B., & Varoquaux, G. (2013). *API design for machine learning software: Experiences from the scikit-learn project*. https://doi.org/10.48550/ARXIV.1309.0238

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297. https://doi.org/10.1007/BF00994018

# Appendix