**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

**Data Cleaning and Preparation**

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

**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**

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

**Conclusion and Recommendations**

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

**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