Classification of Human Activities using Smartphone Sensors through Machine Learning and Neural Network Algorithms

By: Amjad Altuwayjiri and Yongxin Luo

Professor: Amir Jafari

Subject: Machine Learning 1

28 April 2023

1. Introduction

Human activity recognition using smartphones is a growing area of research with many potential applications, including health monitoring, sleep tracking, and fitness tracking (Chawla, Prakash, & Chawla, 2021; Islam et al., 2022). In this project, we used the UCI Human Activity Recognition with Smartphones dataset to train and test machine learning algorithms for accurately classifying different human activities based on accelerometer and gyroscope data collected from a Samsung Galaxy S II smartphone.

Various neural network and machine learning algorithms were applied, including MLP, LVQ, and SVM, to classify the activities and tested their performance using different metrics. The analysis results indicate that the MLP model achieved the highest accuracy of 94%, then LVQ 92% and SVM 91%.Thus, we can conclude that human activities can be accurately classified using accelerometer and gyroscope. However, we argue that this approach can be more effective in a larger and more diverse range of subjects with different health records to detect abnormalities to be useful for detecting unusual behaviors.

2. Description of the dataset

According to Reyes-Ortiz et al. who collected and preprocessed the data, it comes from a group of 30 people between the ages of 19 and 48(2016). They were asked to perform six different activities, three of which involved staying still (standing, sitting, lying down) and three of which involved movement (walking, walking downstairs, walking upstairs) as shown in Figure 1.

Figure1

Activity frequency in the dataset

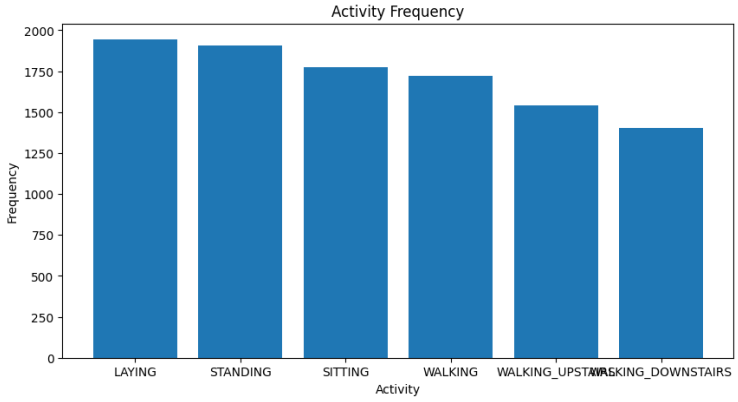


Figure 1: Six activities and their frequency in the dataset

Each participant wore a smartphone on their waist that captured data from the accelerometer and gyroscope 50 times per second as shown in Figure 2 (Reyes-Ortiz et al., 2016). The signals from the sensors were cleaned up and then divided into 2.56-second intervals with 50% overlap, resulting in 128 readings per window (Reyes-Ortiz et al., 2016). From each window, 561 different features were obtained by analyzing the time and frequency and it has 10,299 instances.

Figure2

Two Samples of the Raw Accelerometer and Gyroscope

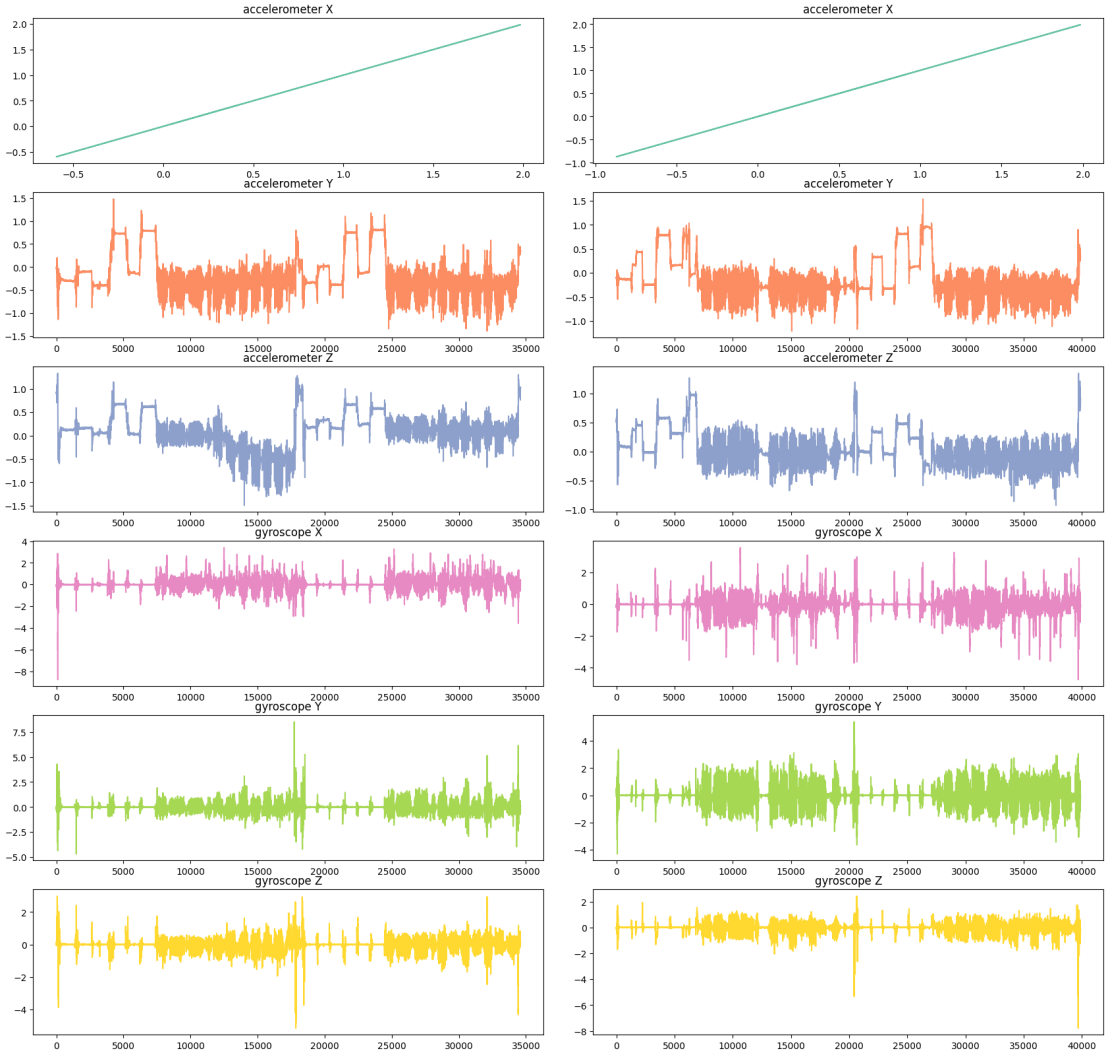


Figure 2: Accelerometer (xyz) and gyroscope (xyz) of two subjects show all six activities

3. Description of the machine learning network and training algorithm

Machine learning network algorithms have become increasingly popular in recent years due to their ability to process large amounts of data and extract useful insights. One of the main

applications of machine learning is in the field of human activity recognition, where algorithms are used to automatically classify human activities based on data from sensors such as accelerometers and gyroscopes (Majidzadeh Gorjani et al., 2021). In this research paper, we focus on four popular machine learning network algorithms for human activity recognition: Multilayer Perceptron (MLP), Learning Vector Quantization (LVQ) and Support Vector Machines (SVM). Each algorithm has its strengths and weaknesses, and we will explore its underlying principles and mathematical formulations. By understanding these algorithms, we can gain insight into how they work and how they can be applied to other machine learning tasks.

3.1. Multilayer Perceptron (MLP)

MLP is a popular type of artificial neural network that was introduced in the 1960s and 1970s by researchers such as Ivakhnenko and Lapa (Ivakhnenko & Lapa, 1966). It is used for classification and regression tasks in machine learning. It is a feedforward neural network, which means that information flows in one direction, from the input layer to the output layer, without any feedback connections (Hagan et al., 2016). Each neuron in the MLP receives inputs from the previous layer, applies a linear transformation to the inputs, and then passes the output through a nonlinear activation function. The backpropagation algorithm is used to train the MLP, which is an optimization method based on gradient descent (Hagan et al., 2016). The MLP equations for each neuron in the hidden layer can be represented as follows:

$$n = w \times p + b$$

$$a = f(n)$$

where w is the weight vector, b is the bias term, p is the input vector, n is the weighted sum of the inputs, f is the activation function, and a is the output of the neuron. The MLP equations for the output layer can be represented as follows:

$$a = \text{softmax}(W \times a + b)$$

where W is the weight matrix, a is the output vector of the hidden layer, b is the bias term, and softmax is the activation function used for classification tasks.

3.2. Learning Vector Quantization (LVQ)

LVQ is an artificial neural network that was first introduced by Kohonenin in 1986 (Kohonen, 1986). It is used for image classification, speech recognition, and bioinformatics, among other applications. LVQ compares the input data to the codebook vectors to determine the closest matching vector or winning unit (Hagan et al., 2016). The codebook vectors are updated to be more representative of the true class distribution in the dataset. The LVQ algorithm consists of initialization, similarity calculation, winner determination, adaptation, and learning rate adjustment steps (Hagan et al., 2016). The adaptation step in LVQ updates the codebook vector based on the input data and the learning rate using the equation:

$$w(q) = w(q-1) + \eta\,(p(q) - w\,(q-1)) = (1-\eta)\,w\,(q-1) + \eta\,p(q)$$

$$w(q) = w(q-1) \quad i \neq i^*$$

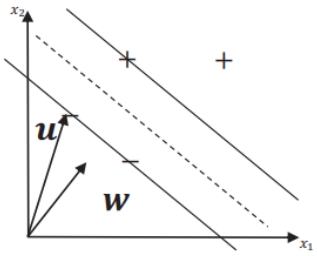
where the term w(q−1) represents the previous value of the codebook vector, while p(q) is the current input vector. The learning rate is represented by η, which controls the amount by which the codebook vector is adjusted. The term (p(q) − w(q−1)) represents the difference between the current input vector and the previous codebook vector. The equation can be rewritten as

(1−η)w(q−1) + ηp(q), which represents a weighted average between the previous codebook vector and the current input vector. The i* in the equation represents the winning neuron, which is the neuron with the smallest Euclidean distance to the input vector.

3.3. Support Vector Machines (SVM)

SVM is a popular supervised machine learning algorithm that was first introduced by Bernhard Boser and his colleagues in 1992 (Boser et al., 1992). It is used for classification and regression tasks in various domains, such as image classification, text classification, and bioinformatics. SVM constructs multiple hyperplanes to separate the data into different classes based on the maximum margin, which is the distance between the hyperplane and the closest data points from each class (Boser et al., 1992). The optimization problem of SVM seeks to maximize the margin between the hyperplane and the closest data points while minimizing the classification error. The optimization problem can be solved using techniques such as quadratic programming and gradient descent.

In SVM, the most important things are to find support vectors and the largest margin. These are the points that are closest to the hyperplane. Margin is the distance between the hyperplane and the observations closest to the support vectors. In SVM large margin is considered a good margin. The largest margin and support vectors are shown below:

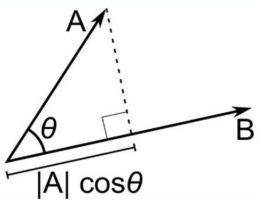


If w.xi + b = 0, it is the equation of margin of SVM. If w.xi + b >= 1, the data point will fall into "+" area, otherwise, the datapoint will fall into "-" area. The formulas are as follow:

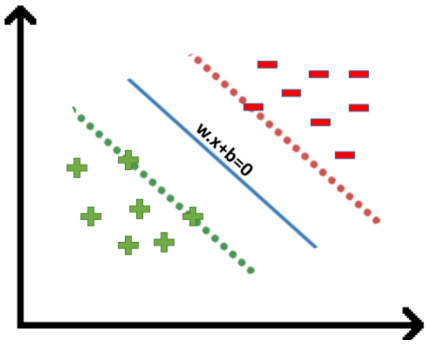
$$\mathbf{w}.\mathbf{x}_+ + b \geq 1$$

$$\mathbf{w}.\mathbf{x}_- + b \leq -1$$

The dot product can be defined as the projection of one vector along with another, multiplied by the product of another vector. The dot product is shown below:



The dot product can be defined as the projection of one vector along with another, multiplied by the product of another vector. The distance of vector w from the origin to the decision boundary is 'c'. Let's say the projection vector of A is B, so if the dot-product of A and B is bigger than C, it is positive samples. If the dot-product of A and B is smaller than C, it's negative samples. The largest margin and "+" / "-" area are shown as follows:



To classify a point as negative or positive we need to define a decision rule. We can define decision rule as:

$$y = \begin{cases} +1 & \text{if } \vec{X}.\vec{w} + b \geq 0 \\ -1 & \text{if } \vec{X}.\vec{w} + b < 0 \end{cases}$$

4. Experimental setup

This part describes the process of using data to train and test a machine learning technique. Specifically, it explains how the machine learning algorithm is implemented in the chosen framework and how its performance is judged.

4.1. Data Preprocessing

The first step in the process is to preprocess the data. The “sounds.csv” file is loaded into a Pandas DataFrame and the feature types, data imbalance, and number of NA values are checked. The next step is to detect outliers. DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is used to perform clustering on the preprocessed data. The algorithm finds points that are closely packed together and mark them as core samples, and the remaining points as noise or outliers. The outliers are then extracted from the data, and a new dataset is created without the outliers for further analysis. The feature matrix X and target vector y are extracted and standard scaling is applied to X to ensure that all the features have similar ranges. Principal

component analysis (PCA) is then used to reduce the dimensionality of the feature matrix X to 20 principal components as shown in Figure 3.

Figure3

Principal component analysis (PCA)

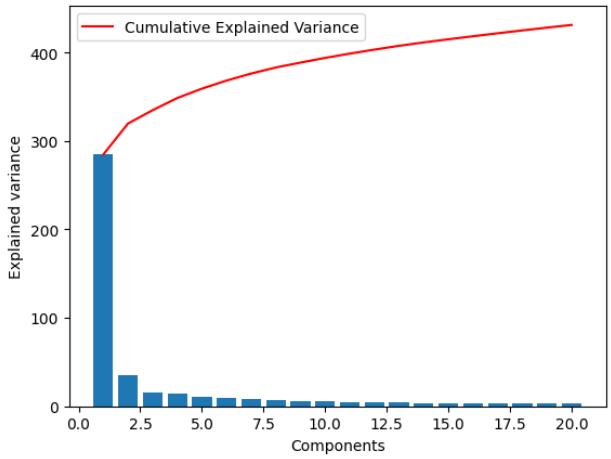


Figure 3: First 20 features selected and their variance

A new data frame is created from the principal components, and the target variable is transformed using label encoding. The preprocessed data is then scaled and split into training and test sets using an 80:20 split.

4.2. Model Training and Evaluation

4.2.1. MLP

The hyperparameters of the MLP classifier are tuned using Bayesian optimization, which tries to find the combination of hyperparameters that maximizes the performance of the model on the training set.

The hyperparameters include the number of hidden layers, the activation function, the solver algorithm, the L2 regularization strength, and the initial learning rate. The hyperparameter search space was defined as follows: the number of hidden layers was sampled uniformly from the range of 1 to 200, the activation function was chosen from a categorical distribution of either ReLU or Tanh, the solver algorithm was also chosen from a categorical distribution of either Adam or L-BFGS, the L2 regularization strength was sampled from a log-uniform distribution ranging from 1e-5 to 1e-3, and the initial learning rate was sampled from a log-uniform distribution ranging from 0.0001 to 0.1.

The best model obtained from the search is (activation = tanh, alpha = 4.950547942793197e-05, hidden_layer_sizes = 144, learning_rate_init = 0.00033522328070073447, solver = adam) and it was evaluated on the test set. We report the accuracy of the model, which is the proportion of correct predictions made by the model. Furthermore, we analyze the performance of the model by computing the confusion matrix, which shows the number of correct and incorrect predictions for each class. Precision, recall, and F1-score are also calculated to provide further insights into the model's performance. Finally, learning curves are plotted to analyze the model's performance with different training set sizes, and the mean and standard deviation of the training and test scores are calculated.

4.2.2. LVQ

The LVQ model is trained and evaluated using several techniques. First, a set of hyperparameters is defined for the model, and a random search is used to find the best combination of hyperparameters for the model by maximizing the performance on the training set.

The parameter distributions were defined to search over a range of possible values, including the number of prototypes per class, the random state, and the beta value. The prototypes_per_class hyperparameter was varied between 1 and 10, while the random_state was set to range between 0 and 9. The beta hyperparameter was varied over a set of discrete values, including 0, 25, 5, 75, and 1.

The best model that is (random_state = 3, prototypes_per_class = 3, beta = 75) evaluated on the test set to calculate the accuracy, which measures the proportion of correct predictions. Additionally, the confusion matrix is computed to show the number of correct and incorrect predictions for each class. The precision, recall, and F1-score are also calculated to provide further insights into the model's performance. Finally, learning curves are plotted to analyze the model's performance with different training set sizes, and the mean and standard deviation of the training and test scores are calculated.

4.2.3. SVM

The SVM model is trained and evaluated using the following code. First, an instance of the SVC model is created with specified hyperparameters, including the penalty parameter of the error term set to be 0.1, a linear kernel function that is used to transform the input data into a higher-dimensional space, and the gamma value that controls the shape of the decision boundary set to be 1. The model is then fitted to the training data. Predictions are made on the testing set

using the trained model, and the accuracy of the model is calculated on both the training and testing sets. The model's accuracy on the training set is printed to the console, followed by its accuracy on the testing set. These accuracy scores measure the proportion of correct predictions made by the model.

5. Results

In this study, we evaluated the performance of three machine learning algorithms: MLP, LVQ, and SVM in terms of their accuracy, precision, recall, and F1-score. The results are presented in Table 1.

Table 1

Evaluation Results of MLP and LVQ

Algorithm	Accuracy	Precision	Recall	F1-score
MLP	94%	0.94	0.94	0.94
LVQ	92%	0.92	0.92	0.92

Table 1: Performance Metrics of the Neural Network Algorithms

As shown in Table 1, the MLP algorithm performed the best, achieving %94 accuracy, precision, recall, and F1-score. LVQ also performed well, with accuracy above 92% and SVM 91%. According to the MLP confusion matrix Figure 4, most of the classes are predicted

correctly but 59 samples of class 1 that is `Standing` are predicted to be `Sitting`, and 48 samples of class 2 that is `Sitting` is predicted to be `Standing`.

Figure4

MLP Confusion Matrix

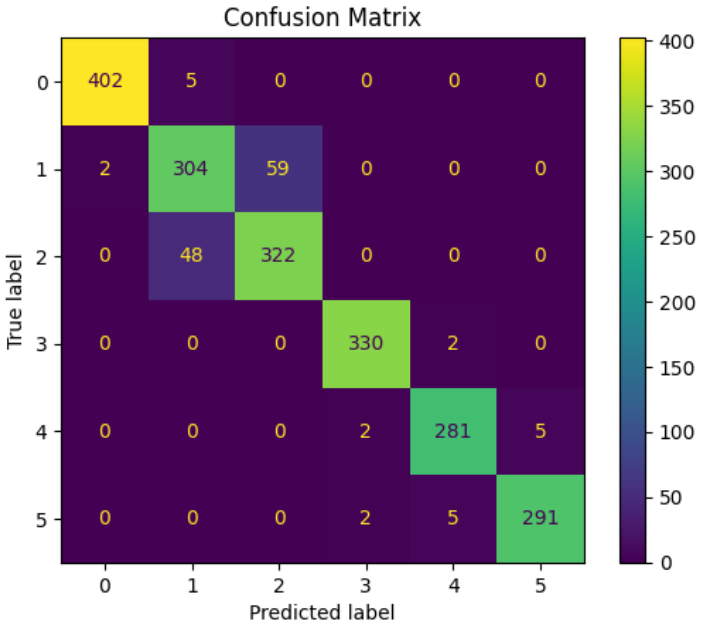


Figure4: Confusion matrix of true and predicted labels of MLP model

The same as MLP, the LVQ confusion matrix Figure 5 shows the same result but with more errors in predicting `Standing` and `Sitting` that it predicted 69 true `Standing` as `Sitting` and 53 true `Sitting` as `Standing`.

Figure5

LVQ Confusion Matrix

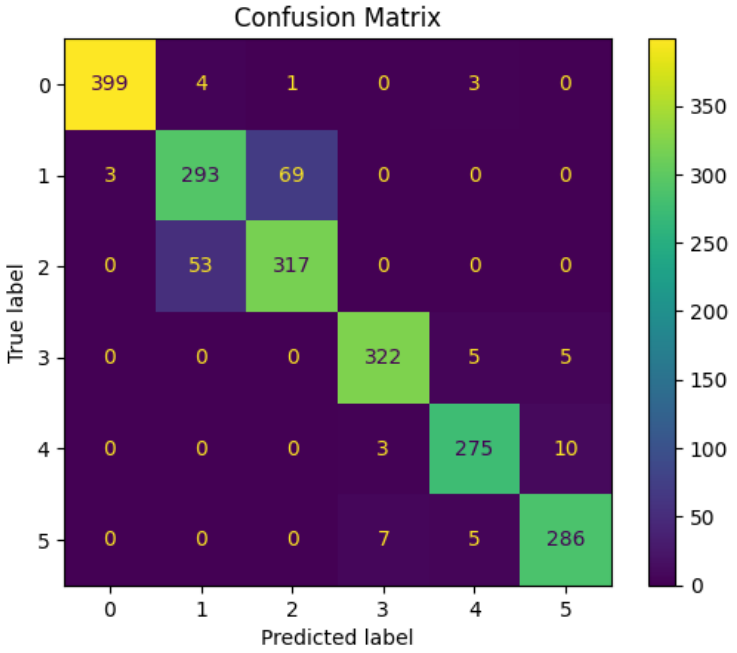


Figure5: Confusion matrix of true and predicted labels of LVQ model

By plotting both models' learning curves, we see that the MLP training curve in Figure 6 is rising with adding more instances to the training, indicating that the MLP model is learning from having more data. However, the training score of 97% is much greater than the validation score of 92% and that indicates that the model requires more training examples or complexity reduction to generalize more effectively.

Figure6

MLP Learning Curve

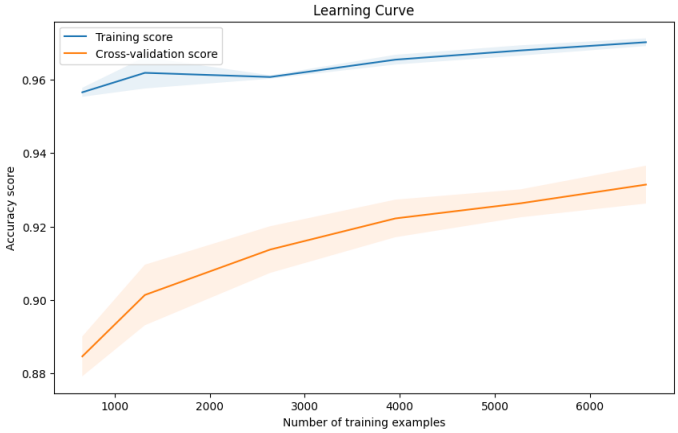


Figure6: MLP training learning curve is raising means getting overfitting

LVQ learning curve Figure 7 is getting lower with adding more instances, but the cross-validation score line got higher, which means that the LVQ model is improving in its generalization ability. The lower accuracy score on the learning curve suggests that the model is becoming less overfit to the training data as more instances are added, while the higher cross-validation accuracy score indicates that the model is performing better on unseen data. This suggests that the model is becoming more robust and reliable in its predictions, and is, therefore, more likely to perform well on new data in the future.

Figure7

LVQ Learning Curve

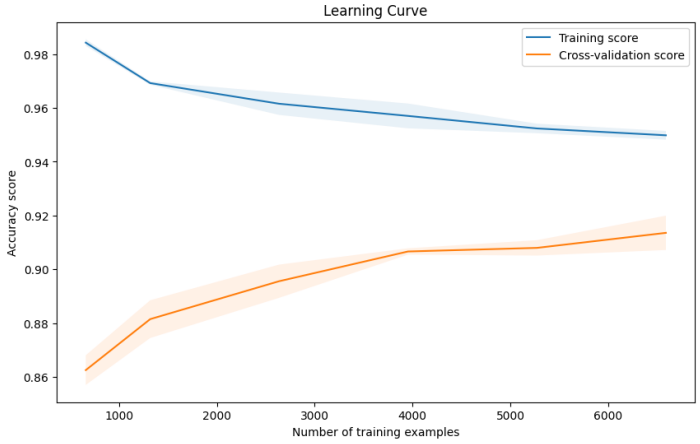


Figure7: LVQ learning curve is steady and steep and not crossing

In LVQ, each neuron in the grid has a weight vector, which is adjusted during training to become similar to the input data points that are closest to that neuron (Hagan et al., 2016). In this way, the SOM learns to group similar data points together and to create a meaningful representation of the input data. A heatmap Figure 8 shows the average distance between each neuron and its neighboring neurons in the grid.

Figure8

SOM Heatmap

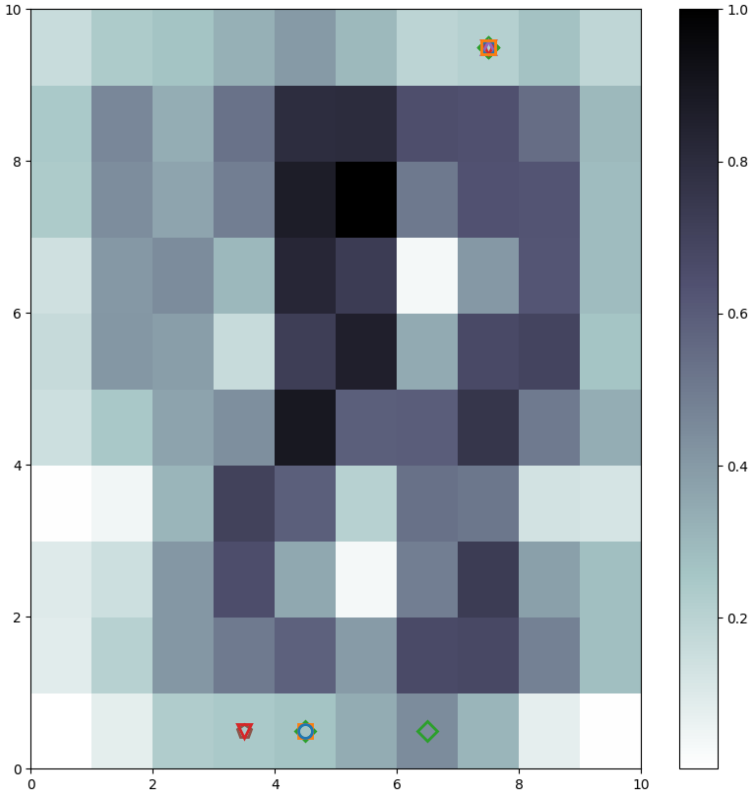


Figure8: Heatmap of the LVQ neurons that are colored based on the distance of other neuron

5.3. SVM

After scaling and doing PCA, we built some SVM models based on cleaned data. As you can see in PCA visualization, our data can be linearly separated. Therefore, basically, we use the linear-svm model as a baseline. And we also built polynomial SVM and sigmoid SVM. Their accuracy, precision, recall rate and f1-score are presented in Table3.

Table 3

Evaluation Results of SVM

Algorithm	Accuracy	Precision	Recall	F1-score	K-fold validation
Linear - SVM	90.92%	0.9114	0.9112	0.9111	0.9092
Polynomial - SVM	93.06%	0.9336	0.9327	0.9328	0.9305
Sigmoid - SVM	76.21%	0.7696	0.7533	0.7526	0.4519

Furthermore, we check if it is overfitting or underfitting. The score of the train set and the test set are shown in Table 4.

Table 4

Overfitting and underfitting of SVM

Algorithm	Score of train set	Score of test set
Linear - SVM	0.9125	0.9092
Polynomial - SVM	0.9439	0.9306
Sigmoid - SVM	0.7642	0.7621

No matter which kernels we chose, the training-set accuracy score and the test-set accuracy are quite comparable. So, there is no question of overfitting or underfitting.

However, it is different from what I discussed earlier. The visualization of support vectors shows the model is overfitting. Anyway, I will fix this problem in the next step.

To fix overfitting, I added early stopping in the Artificial Neural Network.

```
Test loss: 0.0
Test accuracy: 0.17718446254730225
```

I got the loss in the test set is 0, and the accuracy is 0.1771. This accuracy is low, but the overfitting problem is indeed solved. That is because the early stopping monitors the loss rather than the accuracy. The loss quantifies how certain the model is about a prediction and the accuracy merely accounts for the number of correct predictions.

Another solution to fix the overfitting problem is using soft margin in SVM instead of hard one. Soft margin SVM allows some misclassification to happen by relaxing the hard constraints of Support Vector Machine. Soft margin SVM is implemented with the help of the Regularization parameter (C). Regularization parameter (C): It tells us how much misclassification we want to avoid. By lowering the regularization parameter, the margin increases, so the training model is to become a soft SVM. I set all C in different kernels' SVM are no bigger than 50.

Then, to improve the model's performance, I added hyperparameter optimization. The first one is grid search. By setting penalty parameters as 1, 10, 100 and setting kernels as linear, poly, rbf and sigmoid with GridSearchCV method.

The model select *penalty* equals 100 and the *kernel* is to be *RBF*. The information is shown as follow:

```
{'C': 100, 'kernel': 'rbf'}
Accuracy:0.9316659026293929
```

The accuracy is 0.93, and what's worthy mentioned is that it indeed improved from 0.90 to 0.93.

The other one hyperparameter optimization is bayesian optimization with a gaussian process. It selects c equals 42 and the kernel is to be rbf. The information is shown as follows:

```
OrderedDict([('C', 42.42241398290779), ('kernel', 'rbf')])
Accuracy:0.9316657258712034
```

The accuracy is 0.93, which is also improved by 0.03 with bayesian optimization.

6. Conclusions

Machine learning network algorithms such as MLP, LVQ, and SVM can be used to accurately classify human activities by 94% - 92% based on accelerometer and gyroscope data that was collected from a smartphone. The application of these models in real-world scenarios will elevate the health sector, such as using them for health monitoring, sleep observing, and fitness tracking.

One limitation of the dataset used in this study is its small size, as it only includes data from 30 subjects. Additionally, there is a lack of information about the health history of the subjects, such

as physical injuries or obesity, which may limit the generalizability of the findings. We argue that with a larger and more diverse dataset, this approach has the potential to be an effective tool for detecting abnormalities in various populations. Therefore, future studies should focus on expanding the dataset to include a wider range of individuals with different health records to further validate the effectiveness of this approach to detect any unusual behavior that needs immediate intervention.

Reference

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. https://doi.org/10.1145/130385.130401

Chawla, K., Prakash, C., & Chawla, A. (2021). Comparative study of computational techniques for smartphone based human activity recognition. *Lecture Notes in Electrical Engineering*, 427–439. https://doi.org/10.1007/978-981-16-3067-5_32

Hagan, M. T., Demuth, H. B., Beale, M. H., & Jesús Orlando De. (2016). *Neural network design*. s. n.

Islam, M. M., Nooruddin, S., Karray, F., & Muhammad, G. (2022). Human activity recognition using tools of convolutional neural networks: A State of the art review, data sets, challenges, and future prospects. *Computers in Biology and Medicine*, *149*, 106060. https://doi.org/10.1016/j.compbiomed.2022.106060

Ivakhnenko, A. G., & Lapa, V. G. (1966). Cybernetic Predicting Devices. *Cybernetics and Systems Analysis*, *1*(1). https://doi.org/chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://core.ac.uk/download/pdf/41822837.pdf

Kohonen, T. (1986). *Learning vector quantization for pattern recognition*. Helsinki University of Technology.

Majidzadeh Gorjani, O., Byrtus, R., Dohnal, J., Bilik, P., Koziorek, J., & Martinek, R. (2021). Human activity classification using Multilayer Perceptron. *Sensors*, *21*(18), 6207. https://doi.org/10.3390/s21186207

OpenAI. (2023). GPT-3.5 Language Model. [Computer software]. Retrieved from https://openai.com/gpt-3/

Reyes-Ortiz, J.-L., Oneto, L., Samà, A., Parra, X., & Anguita, D. (2016). Transition-aware human activity recognition using smartphones. *Neurocomputing*, *171*, 754–767. https://doi.org/10.1016/j.neucom.2015.07.085

Schölkopf, B., Platt, J. C., Shawe-Taylor, J., Smola, A. J., & Williamson, R. C. (2001). Estimating the support of a high-dimensional distribution. *Neural Computation*, *13*(7), 1443–1471. https://doi.org/10.1162/089976601750264965