Biostat 626 midterm 1 report

Feiyang Deng

University of Michigan, Ann Arbor

Abstract
This report describes the process of developing a machine learning model to predict human activity recognition based on accelerometer data. The model uses a stacking approach with feature augmentation and post-processing using deep neural networks and manual modification to achieve high accuracy on the test set. Furthermore, the report suggests potential further improvements by incorporating RNN-like models to model contextual information.

Keywords
Human activity recognition, machine learning, stacking, feature augmentation, deep neural networks

1. Introduction

1.1. Data description

The Human Activity Recognition dataset contains information on human activities and is used for activity recognition research. The dataset includes 7767 observations in the training data and 3612 observations in the test data. Each observation pertains to a time window and is associated with a subject. The dataset contains a vector of 561 features calculated from the time and frequency domains, and each observation is linked to an activity code from 1 to 12.

For each observation (time window), a vector of 561 features is obtained by calculating variables from both the time and frequency domains. These features represent various measurements and metrics, such as the mean, standard deviation, maximum, and minimum values, as well as the magnitude of signals in three directions. These measurements are obtained from various sensors, including an accelerometer and a gyroscope.

The activities that are performed during the observation period are classified using activity codes ranging from 1 to 12. These activities include walking, walking upstairs, walking downstairs, sitting, standing, and laying down. Additionally, the dataset contains information on other activities, such as standing still while talking on the phone and standing while using a computer.

1.2. Task definition

The objective of Task 1 is to classify activities as either static (activities 4-12) or dynamic (activities 1-3) based on the provided dataset. This is a binary classification task, where the model will be trained to distinguish between these two activity types.

The objective of Task 2 is to classify activities into seven different classes, including walking (activity 1), walking upstairs (activity 2), walking downstairs (activity 3), sitting (activity 4), standing (activity 5), lying (activity 6), and static postural transitions (activities 7-12). This is a multi-class classification task, where the model will be trained to differentiate between these different activity types based on the features provided in the dataset.

1.3. Methods tried in task

In Task 1 and Task 2, various machine learning and deep learning methods were tried to classify the activities in the provided dataset. The machine learning methods include Support Vector Machines (SVM), K-Nearest Neighbors (KNN), Linear Discriminant Analysis (LDA), Naive Bayes, Decision Tree, Random Forest, Gradient Boosting, AdaBoost, and Multi-Layer Perceptron (MLP).

Ensemble methods such as Bagging and Stacking were also attempted to improve the performance of the models. In addition to these methods, deep learning techniques such as 1D Convolutional Neural Networks (CNN1D) and Residual Networks (ResNet1D) were also used.

Data augmentation techniques were also applied to improve the model's performance, including using binary classifier results, lagged features, Synthetic Minority Over-sampling Technique (SMOTE), and Adaptive Synthetic Sampling (ADASYN).

Finally, two types of loss functions were used to optimize the deep learning models: cross-entropy loss and multi-class focal loss. These methods were experimented with in both Task 1 and Task 2, and the best-performing method was selected based on the evaluation metrics.

2. Baseline algorithm

The baseline model for both Task 1 and Task 2 is a bagging model consisting of 10 random forests for a 12-class classification task. The hyperparameters of the random forest were selected using grid search and cross-validation.

The model uses the 12-class classification output and then aggregates the results to classify activities into 2 classes for Task 1 (static and dynamic) and 7 classes for Task 2 (walking, walking upstairs, walking downstairs, sitting, standing, lying, and static postural transitions).

The random forest algorithm achieved 100% accuracy on the training data, but in 10-fold cross-validation, the model achieved an accuracy of 0.982. However, the model's performance decreased when tested on the independent test set, with an accuracy of 0.997 for Task 1 and 0.929 for Task 2. This indicates that Task 1 is relatively easier and the model may be overfitting to the training data and is not generalizing well to new data.

Further exploration of other models and optimization techniques may be necessary to improve the performance of the model on the test set.

3. Final algorithm

The final model for Task 1 is a stacking model that uses K-Nearest Neighbors (KNN), Linear Discriminant Analysis (LDA), Support Vector Machine with linear kernel (SVM-linear), and Support Vector Machine with polynomial kernel (SVM-poly). The model achieved 100% accuracy on both the training and test data.

For Task 2, the final model consists of a stacking model ensemble with multiple machine learning algorithms, including SVM, KNN, LDA, Naive Bayes, Decision Tree, Random Forest,

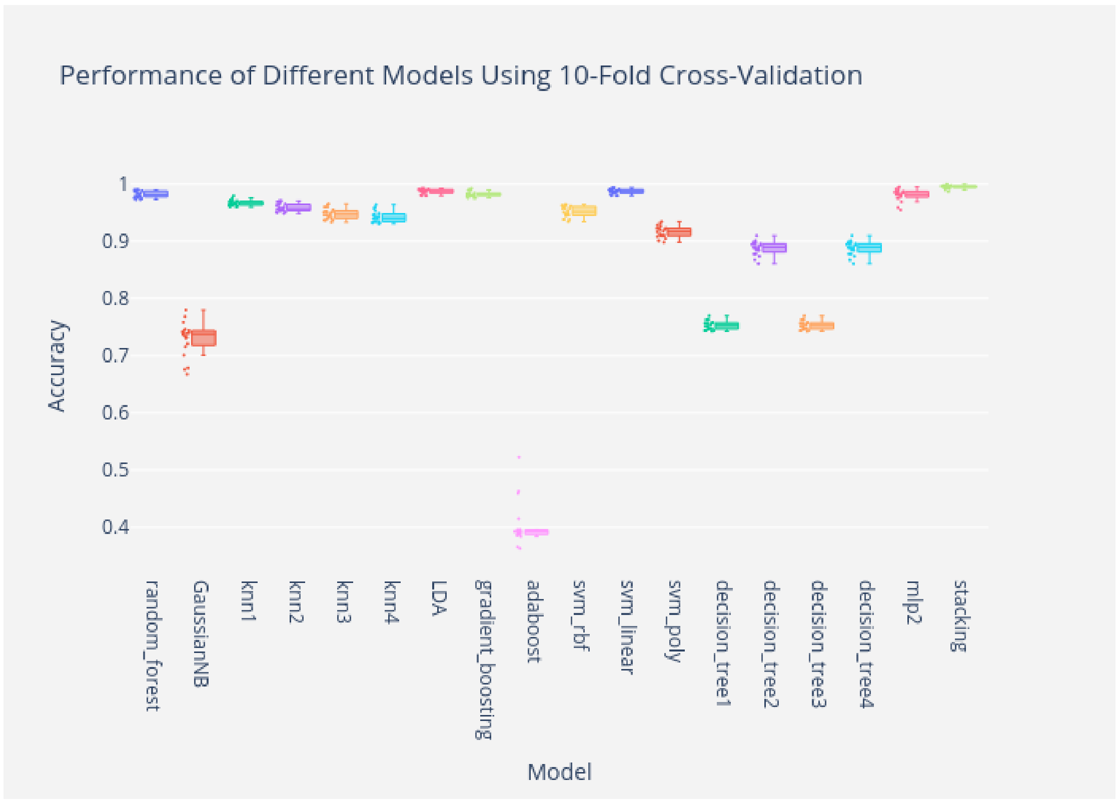

Figure 1: Stacking model cross-validation performance

Table 1
Confusion matrix

		Predicted						
		1	2	3	4	5	6	7
Actual	1	1226	0	0	0	0	0	0
	2	0	1073	0	0	0	0	0
	3	1	0	986	0	0	0	0
	4	0	0	0	1269	21	0	3
	5	0	0	0	15	1408	0	0
	6	0	0	0	0	0	1412	1
	7	0	1	0	1	0	3	347

Gradient Boosting, AdaBoost, and MLP. The model also incorporates additional binary and 2 lagged features. This stacking model achieved 100% accuracy on the training data and 94.4% accuracy in 10-fold cross-validation. The performance of different models in the stacking model is shown in Figure 1.

However, the confusion matrix Table 1 showed that most of the misclassifications were between activities 4 and 5, especially misclassifying 4 as 5. To address this issue, four deep neural networks were trained to classify activities 4, 5, and others using 1D convolutional

Table 2
Leader board performance

Algorithm	Performence
12-class classification Rf + bagging	0.929
7-class classification stacking	0.962
Stacking+Binary+Lag1	0.967
Stacking(mlp)+Binary+Lag1	0.971
Stacking(mlp)+Binary+Lag2	0.976
Stacking(mlp)+Binary+Lag2+CNN1D+ADAYSN	0.977
Stacking(mlp)+Binary+Lag2+CNN1D+Resnet1D+ADAYSN	0.980
Stacking(mlp)+Binary+Lag2+CNN1D+Resnet1D+ADAYSN+Focal loss+Manual	**0.987**

neural networks (CNN1D) and residual networks (ResNet1D). These models were trained on data generated by Adaptive Synthetic (ADASYN) sampling technique and optimized using cross-entropy loss and multi-class focal loss.

Based on the original predictions from the stacking model, if a sample was predicted as class 5, it was reclassified as class 4 if the neural network predicted the sample as class 4 with high probability, and if there was only a small number of such disagreements. This post-processing step helped to improve the accuracy of the model and alleviate the misclassification issue between activities 4 and 5.

Finally, by observing the distribution characteristics of the predicted behaviors in the test set, it was found that behaviors 1-6 always showed a tendency to appear together, while behavior 7 appeared only at the joining point of the interval where behavior 1-6 changed (actually 4-6). To exploit this distribution, all the predictions in the test set were manually examined and obvious errors were fixed, resulting in a significant improvement of 0.007 on the test set.

4. Building process

This section presents a detailed analysis of the problem and the model building process, along with the challenges faced during training and the techniques employed to overcome them. The results obtained are summarized in Table 2.

4.1. Stacking model

Initially, a 12-category classification task was performed by integrating Task 1 and Task 2 using the random forest and bagging methods. A high accuracy of 0.997 was achieved for Task 1, but the accuracy for Task 2 was lower than expected, reaching only 0.929, which was significantly lower than the cross-validation score of 0.98 on the training set. This could be attributed to the low number of categories 7-12 out of 12, and therefore, it was deemed more appropriate to treat 7-12 as a single category in the training process for Task 2.

In order to improve the model's performance, a stacking integration method of different machine learning algorithms was tried, and it resulted in a significant improvement compared to the bagging integration of a single model. This approach allowed for the integration of several models to make predictions, and the final prediction was made by a meta-model using the predictions from the base models as inputs.

4.2. Feature augmentation

As the data is collected continuously for each subject, it is reasonable to consider the connection between each time window and the following time window. Therefore, feature augmentation was used by concatenating the features of the current window and the previous window (Lag1) to predict the current activity state. This approach resulted in a 0.002 improvement in cross-validation, indicating that adding lag features was effective. Additionally, concatenating the feature of the previous 2 windows (Lag2) resulted in even better performance in cross-validation.

Furthermore, the binary classification results were also used as an additional feature. Using the stacking model, a full accuracy of Task 1 binary classification was achieved on the test set. Based on these feature augmentation techniques, the current machine learning stacking model achieved a high accuracy of 0.976 on the test set after cross-validation to select hyperparameters.

Overall, the stacking model with feature augmentation proved to be an effective approach for achieving high accuracy in the classification of activities.

4.3. Deep learning post-processing

Sklearn provides support for MLP (Multi-layer Perceptron) neural networks, and adding MLP to the stacking model improved test set performance by 0.05. However, the confusion matrix from cross-validation showed that misclassifications mostly occurred between class 4 and 5, with class 4 often misclassified as class 5. Although the stacking model achieved high accuracy on the test set (0.976), efforts were needed to address the misclassifications between class 4 and 5. To address this issue, deep neural networks were used to further improve the model's performance.

This task was treated as a 3-class classification problem (class 4, class 5, and others). However, the number of samples in classes 4 and 5 was relatively small (about 1000) compared to the 5000 other samples, creating an unbalanced situation. To address the imbalance, SMOTE and ADAYSN (Adaptive Synthetic) were used to oversample classes 4 and 5. ADAYSN proved to be more effective in balancing the classes, as demonstrated by cross-validation results on the stacking model.

Initially, a deep 1D CNN network was used in conjunction with ADAYSN to correct the predictions of the stacking model. A validation set comprising 20% of the training set was used to select hyperparameters and training epochs. To ensure that the CNN network produced high-quality predictions for class 4 and 5, it was checked that the results were mostly consistent with the stacking model, with the CNN network more likely to output class 4. Disagreements between the two models were resolved only when the CNN network predicted class 4 with high probability (>0.8) and the stacking model predicted class 5. The number of fixed predictions was limited to a maximum of 10 to avoid introducing additional errors. This fix improved the test set performance by 0.001, indicating that the CNN network was effective.

To further increase model complexity, the Resnet 1D network was used for correction on top of the original model. The same criterion was used for selecting hyperparameters and training

Table 3
Misclassification example

ID	Prediction
1524	4
1525	4
1526	4
1527	5
1528	4
1529	4
1530	4
1531	4

epochs, and the resulting model achieved a test set accuracy of 0.980.

Analysis of the results showed that the model struggled most with distinguishing between classes 4 and 5. To further address this, multi-class Focal loss was used to train the models on unbalanced samples and samples with classification difficulties. By retraining both models using multi-class Focal loss, the models converge faster than using cross-entropy loss. Overall, the post-processing with deep learning achieved substantial improvements in model performance.

4.4. Manual post-processing

With an achieved accuracy of approximately 0.98 on the test set, we have a considerable level of confidence in the current prediction results. Upon analyzing the distribution of the prediction results on the test set, we have observed that the activity of the subjects exhibits a clear continuous character, as the data are consistently ordered. This means that the subjects can only maintain the same activity during a series of consecutive time windows, resulting in the occurrence of the same type of activity continuously. Activity 7 appears only at the connection where the activity changes.

Interestingly, we have identified evident classification errors in the prediction results, such as a prediction of 5 in a context where all other predictions were 4 in Table 3. Due to time constraints and the difficulty of defining rules, we decided to correct these errors manually. The manual correction of these errors ultimately resulted in a notable improvement of 0.007 on the test set.

5. Potential further improvement

The current model achieved high accuracy through the inclusion of lagging features in the modeling process. However, since each observation in the data can be treated as a continuous window, it is reasonable to consider using RNN-like models to model contextual information. Models such as RNN, LSTM, and Transformer are suitable for capturing temporal dependencies in sequential data, and may provide further improvements in performance.

To train these models, a dataset can be obtained by sampling several consecutive windows for each subject. For example, 10 consecutive windows can be sampled for each subject to obtain a dataset that captures the temporal dependencies of the data. With this dataset, RNN-like models can be trained to capture the temporal dynamics of the data and potentially improve the performance of the current model.