Sleep Dataset Analysis Report

Ruixuan Dong

April 23, 2023

---

1 Packages

Begin by importing the packages we'll need during this assignment.

1. numpy is the fundamental package for scientific computing with Python.
2. pandas is the package for operating data frame with Python.
3. matplotlib is a library to plot graphs in Python.
4. tensorflow is a library for machine learning and artificial intelligence.
5. `utilities` provides some functions to implemente data set.

2 Load and Process the Dataset

We'll be using the "Sleep toy" dataset in this study. There are 1000 columns in total, where the first column represents a time stamp, the second column `Action` is set as label, and the last 998 columns are features in this case. We'll try to use features to establish an efficient classifier and try to predict actions while sleeping much better. In the first step, we choose the first 70% of each day's(in time order) rows of the original data set as our training set, and keep the rest 30% rows as our testing set.

After selecting training set and testing set, we tried to randomized the order of each set using function `shuffle` in `random` package.

Next, the dimension of each dataset is displayed in the table below. The dimension of dataset is one of the most important thing during the analysis step, and `.shape` is used to obtain the dimension of each dataset.

Table 1: Dimension of Each Data Set

Dataset	Shape
train_set_x_orig	(2516, 998)
train_set_y_orig	(2516,)
test_set_x_orig	(1079, 998)
test_set_y_orig	(1079,)
whole_set	(3595, 1000)

In order to know our data set better, we also need to check how many kinds of label are included in the sleeping actions, name of each sleeping action and counts of them. Here is a table displaying these information:

Table 2: Counts for Different Sleeping Actions

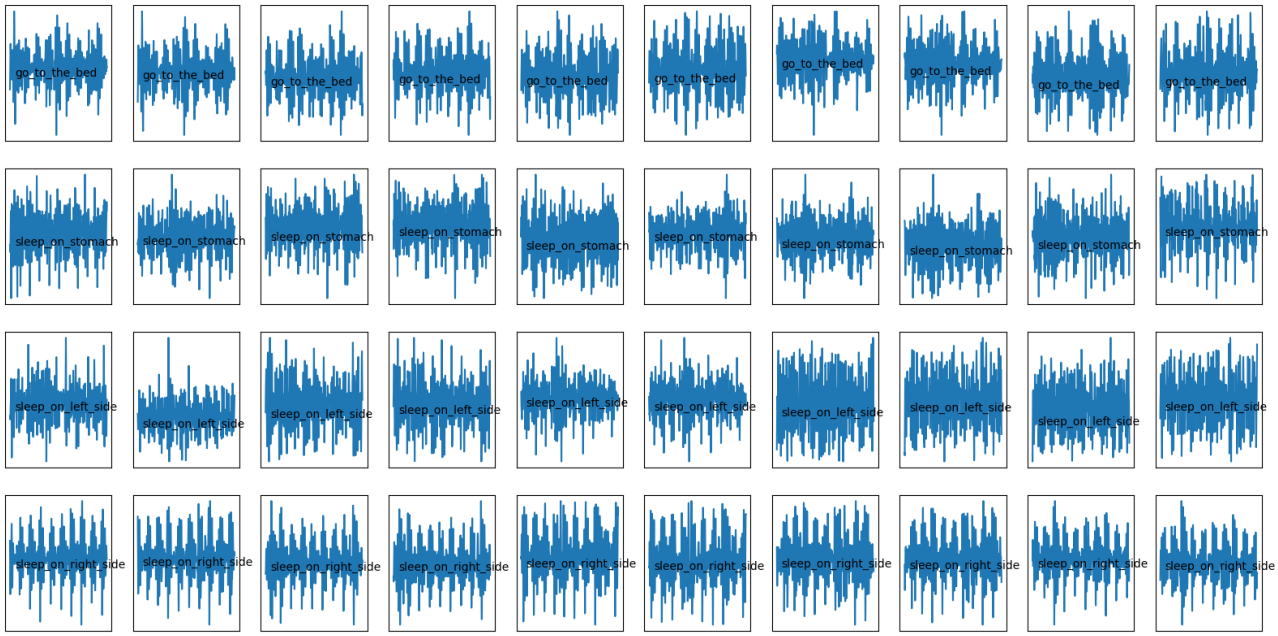
Action	Counts
go_to_the_bed	2974
sleep_on_stomach	390
sleep_on_left_side	138
sleep_on_right_side	93

To make it easier to establish model, we change the label from object type into int type.Here are an explanation of how we represent the labels. These are the original actions during sleeping period, and we also present how to convert them to new labels. Below are the transformed rules:

Table 3: Convert Sleeping Actions to Numbers

Action Type (object)	Number (int64)
go_to_the_bed	0
sleep_on_stomach	1
sleep_on_left_side	2
sleep_on_right_side	3

The plot below shows how time series plots differs based on the different sleeping actions.



3 Model Architecture

In this part, we're going to establish neural network model for classification work. First, we'll try a common neural network and check the performance of it.

3.1 One-Hot Encodings

This study in deep learning we have a y vector with numbers ranging from 0 to C-1, where C = 4 is the number of classes here. Then in this case, C = 4, then we have the following y vector which will be needed to convert as follows:

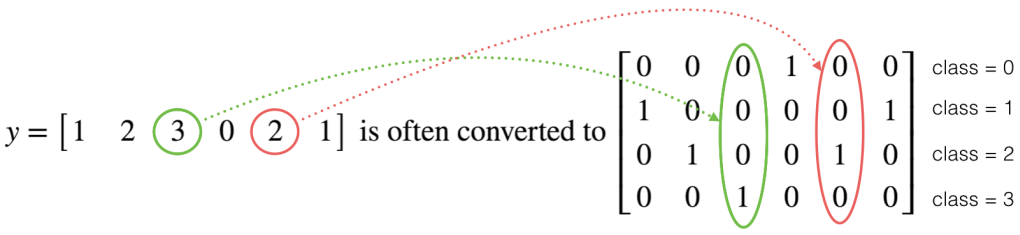


Figure 1: Time series plots for different sleeping actions

This is called a "one hot" encoding, because in the converted representation exactly one element of each column is "hot" (meaning set to 1). To do this conversion in pandas, we can

use one line of code:

- pd.get_dummies(labels)

3.2 First Neural Network

In this part, we're going to build our first neural network in this study. Before building neural network, we need to normalize the features of training set.

Now we try to define some criterias, like recall rate, precision rate and F-1 score. After that, we create the neural network using `tf.keras.Sequential()`. Here is some details about this neural network:

1. The input layer has a dimension of $m \times 998$, where m is the sample size and 998 is the number of features.
2. The first hidden layer has 1000 neurals with 50% dropping out, which is implemented here for avoiding overfitting, and the activation function is ReLu.
3. The second hidden layer has 1000 neurals with 50% dropping out and activation function as ReLu.
4. The third hidden layer has 200 neurals with 30% dropping out and activation function as ReLu.
5. The fourth hidden layer has 20 neurals with 20% dropping out and activation function as ReLu.
6. The output layer has 4 neurals since this case is a 4-classes classification problem, and we set the activation function as softmax.

Some `callbacks` are set for using early stopping and avoiding overfitting.

The results from our first neural network seems like the training accuracy is much lower than human's performance, having an accuracy only around 0.90, which means our first neural network is not enough and we need a larger one.

3.3 Larger Neural Network

To get a larger neural network, we try to add layers, reduce dropping out proportion and add neurals into some layers, and the new larger neural network is shown below:

1. The input layer has a dimension of $m \times 998$, where m is the sample size and 998 is the number of features.

2. The first hidden layer has 1000 neurals with 30% dropping out, which is implemented here for avoiding overfitting, and the activation function is ReLu.

3. The second hidden layer has 1000 neurals with 30% dropping out and activation function as ReLu.

4. The second hidden layer has 1000 neurals with 30% dropping out and activation function as ReLu.

5. The third hidden layer has 100 neurals with 50% dropping out and activation function as ReLu.

6. The fourth hidden layer has 20 neurals with 20% dropping out and activation function as ReLu.

7. The output layer has 4 neurals since this case is a 4-classes classification problem, and we set the activation function as softmax.

After checking the performance, we can get that there is a narrow bias between human performance and the training set performance (with accuracy around 0.96), which means adding a hidden layer, more neurals into our first neural network does improve the performance on training set. However, the difference between the validation set (around 0.85) and training set (around 0.96) accuracy is still huge, which strongly indicates a overfitting problem, since the validation set and the training set are from the same distribution. Therefore, we need to consider several ways to solve this problem:

1. Get more training data.

2. Regularization (L2, Dropout, data augmentation).

3. Find better NN architecture/hyperparameters search.

After trying L2 penalized model, we found that the imporvement is slight. Besides, getting more data is not valid in this case. Then, we try to implement data augmentation.

3.4 Data augmentation

Data augmentation is widly used in processing image data set, with flipping or rotation, the data set can be added noises or changed window slices, which is a technique of artificially increasing the training set by creating modified copies of a dataset using existing data.

These are different kinds of data augmentation methods which will be implemented in our study later:

1. jittering: Adding jittering, or noise, to the time series.
2. scaling: Scaling each time series by a constant amount.
3. permutation: Random permutation of segments.
4. magnitude_warp: the magnitude of each time series is multiplied by a curve created by cubicspline with a set number of knots at random magnitudes.
5. time_warp: Random smooth time warping.
6. rotation: For 1D time series, randomly flipping.
7. window_slice: Cropping the time series.
8. window_warp: Randomly warps a window by scales.
9. spawner: Based on: K. Kamycki, T. Kapuscinski, M. Oszust, "Data Augmentation with Suboptimal Warping for Time-Series Classification," Sensors, vol. 20, no. 1, 2020.

Now check the figure below see how different kinds of data augmentation can change the original data set. Orange lines represents data after augmentation, while blue lines shows the original data.

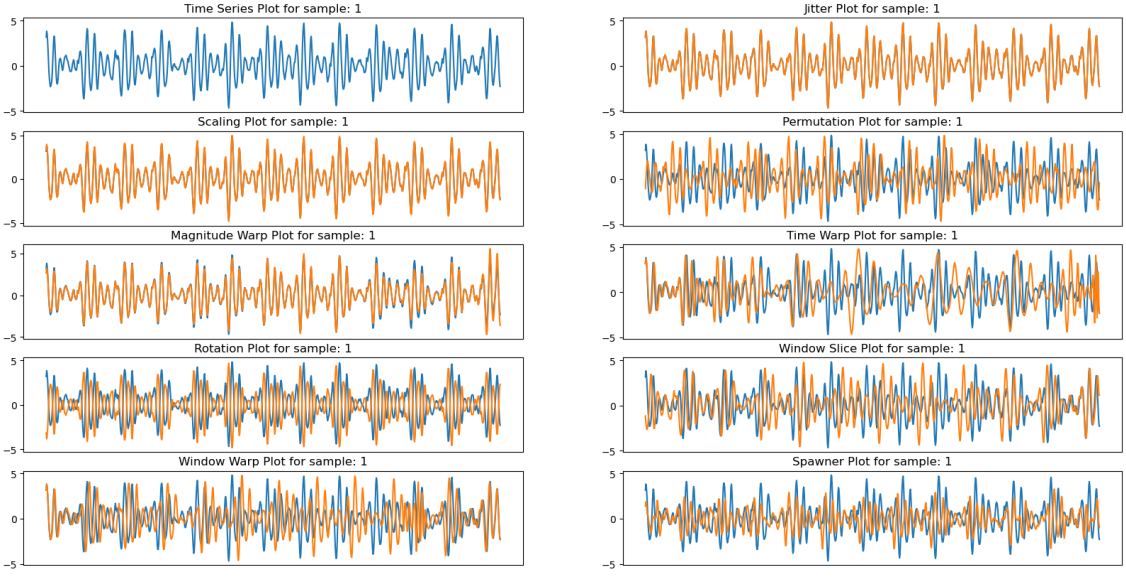


Figure 2: Time series plots for different sleeping actions

It's really excited to see such an improvement of performance on validation set after implementing data augmentation(increasing from 0.85 to 0.89)! Besides, the larger neural network fitted by augmentation data set is stored in 'my_model.h5', which could be found in this folder. In addition, the accuracy of training set also increases. Although there still exists a little overfitting problem, it's much better now. Now let's asses our classifier on the testing set.

Table 4: Prediction Results for Different Classifiers

Classifier	Accuracy
first_NN	0.9604
Larger_NN	0.9245
Larger_NN with data augmentation	0.9622

4 Error Analysis

The confusion matrix of larger neural network with data augmentation is given below:

Table 5: Prediction Results for Different Classifiers

Predicted Label	True Label			
	go_to_the_bed	sleep_on_stomach	sleep_on_left_side	sleep_on_right_side
go_to_the_bed	1044	6	0	0
sleep_on_stomach	20	1	0	0
sleep_on_left_side	12	1	0	0
sleep_on_right_side	2	0	0	0

5 Reload Data Set

In this part, we try to use a different way to create training and testing data: Each time use 14 day's of data as training and set the remaining 1 day as testing. To achieve this goal, we set use `create_larger_model()` as framework and fit models based on each training set, i.e. we'll fit 15 neural network in total in this part. Then, we will use each day's data as testing set to assess each NN model.

Each fitted model is saved in an ".h5" file, which can be found under this folder. Then, the prediction results for each testing set are displayed below:

Table 6: Prediction Results for Different Classifiers

	loss	accuracy	f1_score	precision	recall
Day1	2.895551	0.616000	0.620151	0.620151	0.620151
Day2	0.153497	0.934426	0.933894	0.933894	0.933894
Day3	3.180357	0.678832	0.680941	0.680941	0.680941
Day4	7.280731	0.250000	0.328125	0.328125	0.328125
Day5	0.759509	0.914894	0.884375	0.884375	0.884375
Day6	1.551494	0.782609	0.778977	0.778977	0.778977
Day7	0.966241	0.847896	0.848214	0.848214	0.848214
Day8	0.919156	0.854545	0.864955	0.864955	0.864955
Day9	4.231585	0.554745	0.554861	0.554861	0.554861
Day10	0.940098	0.816092	0.798699	0.801658	0.795833
Day11	0.353538	0.929612	0.929067	0.931352	0.926854
Day12	1.722895	0.754717	0.757143	0.757143	0.757143
Day13	0.511331	0.870968	0.877778	0.884946	0.870833
Day14	2.206282	0.772727	0.772236	0.772236	0.772236
Day15	0.866155	0.878519	0.869792	0.869792	0.869792

6 Conclusion

Based on our study results, we found that there is a mislabeling problem in both the two ways of slicing data set (one for setting every day's first 70% as training set, another for setting the other 14 day's data as training set). One of the most possible cause for this mislabeling problem is training set and testing set are from different distribution. For example, considering the first data slicing case, there are only 8 cases are labeled as "sleep_on_stomach" and even no "sleep_on_left_side" and "sleep_on_right_side". This could be seen as a reason for the mislabeling problem shown in Table 4.