Machine learning in EMG-based gesture recognition

ZHANG Yan

February 1st, 2020

Abstract

The electromyographic signal-based static gesture classification problem was studied using different machine learning algorithms including mainly the Principal Component Analysis and Support Vector Machine. Python scripts were writen to acquire the data and extract features from them, visualize the data by the first two principal components, and develop a classifier to achieve online classification.

1 Introduction

The electromyographic signal (EMG) is a signal that comes out from our muscles during their contractions [1] and contains the information to predict people's intent such as which movement the person wants to perform. Machine learning (ML) is a set of methods to extract the information from data directly without deep understanding of the knowledge behind the data. It is interesting to study the EMG-based gesture classification problem using different ML algorithms like the Principal Component Analysis (PCA), Support Vector Machine (SVM) and Deep Neural Network (DNN).

There are two stages of the project:

1. Build a basic system which includes tree parts: data acqusition and preprocessing, classifier training, and evaluation system. In this stage, static gesture classification problem will be researched using mainly PCA and SVM.

2. Study the dynamic gesture classification problem, where the state-of-art learning algorithm DNN will be applied to draw features from the raw data.

We will describe the details of our solution in the first stage in Section 2, then present some preliminary results in Section 3, finally giving a conclusion and a plan for the second stage in Section 4.

2 Methods

2.1 Data acquisition

2.1.1 Equipment

In this project, the Myo armband developed by Thalmic Labs Inc. was used. This device can return eight surface electromyogram signals with eight bit precision at frequency of 200Hz (Fig. 1 (a)).

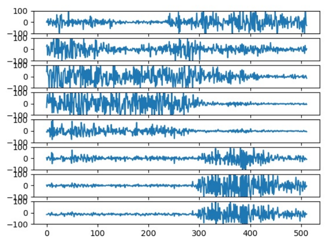
Besides,

- MyoConnect.exe, the software to connect the armband,
- *myo-python*, a python package for working with the armband,
- Myo SDK, which allows the python package to communicate with the Myo hardware through bindings to the *libmyo C* library,

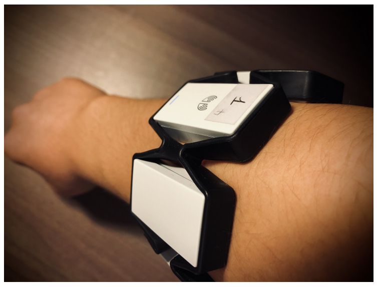
as well as PyCharm and a laptop were used.

2.1.2 Data acquisition

During signal acquisition, the subject sited on a chair wearing the armband on the right forearm (Fig. 1 (b)) and kept the angle at the elbow joint at approximately 90°. Meanwhile, the plot of signals like Fig. 1 (a) was shown to visualize the data being collected.



(a) An example of 8-channel surface EMG signal.



(b) Positioning of the armband on the subject's hand.

Figure 1: (a) shows the eight EMG signals; (b) is the position of the Myo armband on the right arm.

Six hand gestures were chosen for classification in this project (Fig. 2): up, down, left, right, open and close plus rest, having seven motion classes in total. For each gesture, 20 seconds of raw data were obtained and labeled for training of the classifier, which means 28000 $(20 * 200 * 7)$ samples totally.

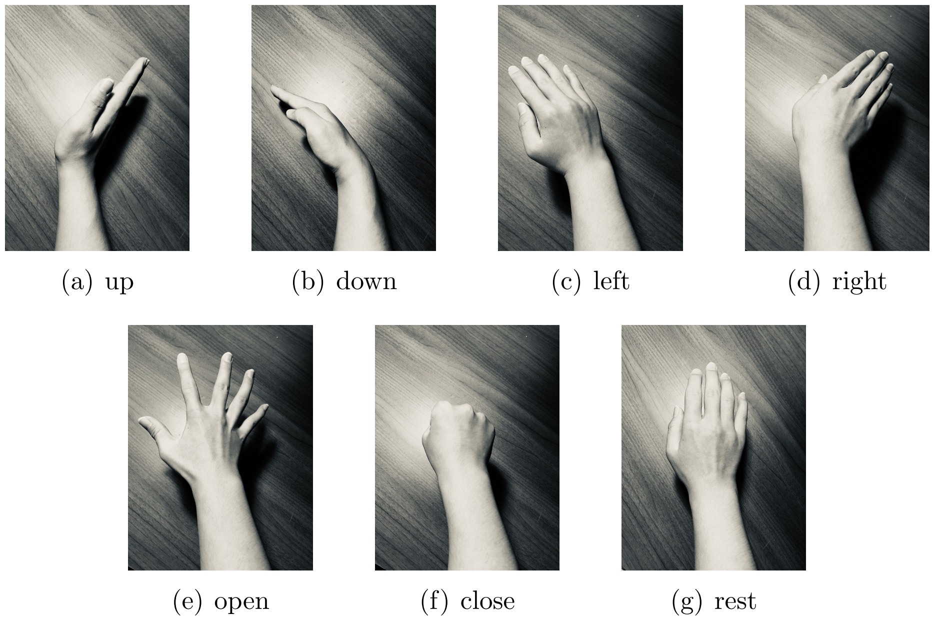


Figure 2: There are seven classes of gestures to classify.

2.2 Feature extraction

Before classification, to reduce dimensionality and extract valuable information of EMG, a process called feature extraction was introduced as well as the corresponding six kinds of features [2]. Say we have a signal $x = \{x_k,\ k = 1, 2, ..., N\}$ in a window, set a threshold $T > 0$ and give a positive integer s, all these features are calculated as following:

Mean Value:

$$MV = \frac{1}{N}\sum_{k=1}^{N} x_k;$$

Standard Deviation:

$$SD = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(x_k - MV)^2};$$

Waveform Length:

$$WL = \sum_{k=1}^{N-1} |x_{k+1} - x_k|;$$

Zero Crossing:

$$ZC = \sum_{k=1}^{N-1} I_{ZC}(x_k),$$

$$\text{where } I_{ZC}(x_k) = \begin{cases} 1 & \text{if } x_{k+1} * x_k < 0 \text{ and } |x_{k+1} - x_k| > T, \\ 0 & \text{otherwise;} \end{cases}$$

Slope Sign Change:

$$SSC = \sum_{k=2}^{N-1} I_{SSC}(x_k),$$

$$\text{where } I_{SSC}(x_k) = \begin{cases} 1 & \text{if } (x_k - x_{k-1}) * (x_k - x_{k+1}) > T, \\ 0 & \text{otherwise;} \end{cases}$$

Autoregressive Coefficients:

$$AC = (X^T * X)^{-1} * (X^T * y),$$

$$\text{where } X_{ij} = x_{i+j-1},\ y_i = x_{i+s};\ i = 1, 2, ..., (N - s),\ j = 1, 2, ..., s.$$

2.3 Data structure and PCA visualization

2.3.1 Training data

Based on the feature extraction methods, the training data can be drew from labeled samples of seven gestures. We let $N = 25$, $T = 3$ and $s = 2$, so the training data with dimensionality of 56 ((5+s)*8) was obtained. Since only the classification of static gestures was discussed, via window overlapping (Fig. 3), 3976 labeled data was obtained for each gesture, which equals to 4000 minus N and plus 1.

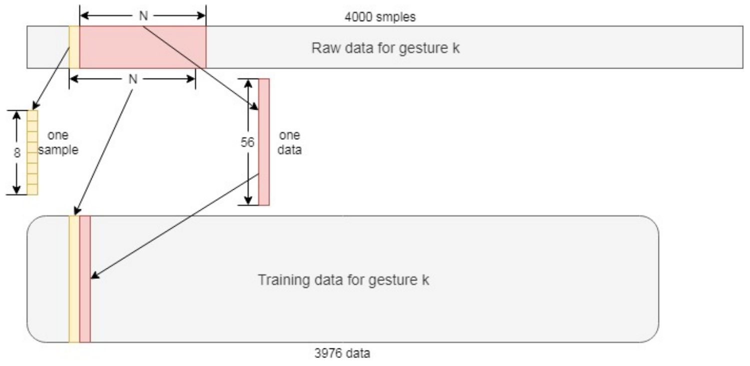
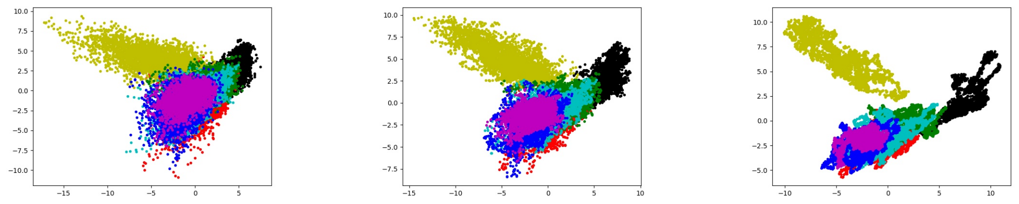


Figure 3: This figure shows the transform from the raw data to the training data by feature extraction via window overlapping.

2.3.2 PCA visualization

In addition to dimensionality reduction, PCA can also give us a good sense about the structure of data by data visualization, so that we can find the most appropriate classification model. Since the first two principal components are superior than the rest, some 2D-figures were plotted using them with different N (Fig. 4). They show that longer windows lead to denser clusters with clearer boundaries, and some classes such as the rest and the down will be more separable linearly.



(a) $N = 10$ (b) $N = 25$ (c) $N = 100$

Figure 4: These are figures of labeled data based on PCA visualization with different N. (black: rest, red: up, yellow: down, green: left, cyan: right, blue: open, purple: close)

2.4 Data classification via SVM model

2.4.1 Support Vector Machine

Since, according to the Fig. 4, this is more like a non-linear classification problem especially when N is small, support vector machine with Gaussian kernel (Equ. (1)) was applied.

$$J(\theta) = C\sum_{i=1}^{m}[y^{(i)}Cost_1(\theta^T f^{(i)}) + (1-y^{(i)})Cost_0(\theta^T f^{(i)})] + \frac{1}{2}\sum_{j=1}^{m}\theta_j^2, \tag{1}$$

$$\text{where } f_j^{(i)} = exp(-\frac{||x^{(i)} - x^{(j)}||^2}{2\sigma^2}),\ Cost_i(x) = \max(0, 1 + (-1)^i x),\ i = 1, 2.$$

Two hyperparameters are crucial for this model, C and σ. They control the classification margin between different classes and the influential ability of a single data point. We illustrate it by a two-category classification problem in Fig. 5.

2.4.2 Model training

Given training data X and the corresponding label y, before the training, two important techniques were applied to increase the performance and speed up the algorithm. The first one was data normalization, which is shown in Equ. (2).

$$X_i = \frac{X_i - \mu_i}{s_i}, \tag{2}$$

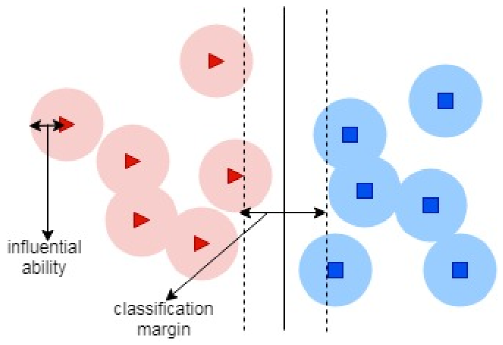


Figure 5: The relationship between the hyperparameters and the classifier is: classification margin $\propto \frac{1}{C}$, influential ability $\propto \sigma$.

where μ_i and s_i are the sample mean and sample standard deviation for the feature i. Then, PCA was used again to reduce the dimensionality of X, and finally we got a training set of 25 component features and 27832 (3976*7) data points.

During the training, data set was splitted into 70% of training set and 30% of test set, and the well-known one-vs-rest technique was chosen to achieve multi-class classification. Since there were just two hyperparameters for SVM which we often need to modify after change some other parameters like N and the time of running the training algorithm one time was not short, C and σ were chosen manually.

Finally, we got a classifier with precision of 100% on both the training and test data sets. It was stored using the *pickle* package with some necessary data and python objects in order to apply them to a new data set.

2.4.3 Online classification

In the end, an online classification script was written which collected the data every 125ms. And to avoid misclassification, a full voting filter was utilised, which may delay the classification by 125ms.

There are two points we should note about the Fig. 6. Firstly, after the feature extraction, normalization should be applied to the data we obtained, but we must use the old sample mean and sample standard deviation we got from the model training. Moreover, only when the predictions in the filter are all the same value can we make sure which gesture it should be, which improves the robustness of the prediction system.

3 Results

In Fig. 7, we can see 90.78% of variance was remained, and on this set of gestures, the features making most contributions were SD, WL and ZC. In Fig. 8, the training result of our SVM classifier on 27832 data is shown.

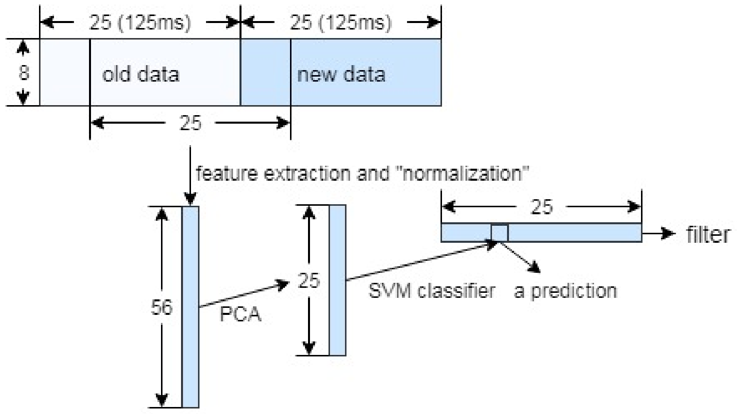
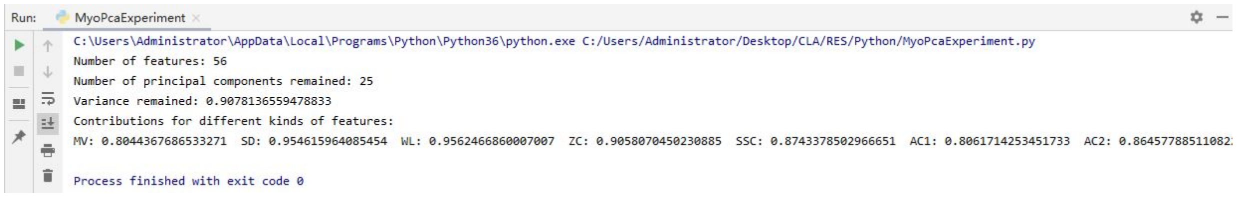


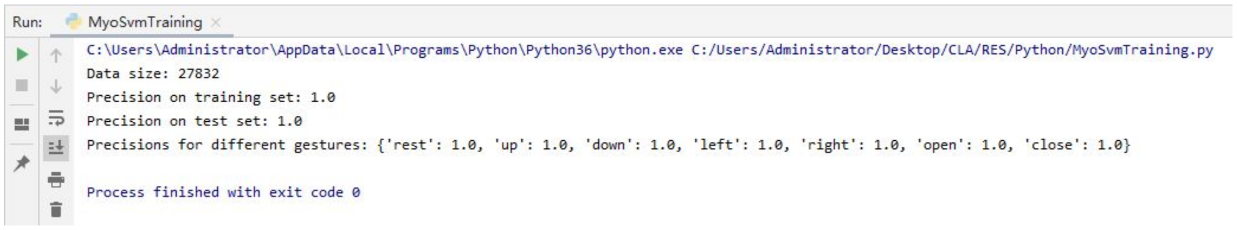
Figure 6: This is the working principle diagram of the PCA classifier with a filter.



```
C:\Users\Administrator\AppData\Local\Programs\Python\Python36\python.exe C:/Users/Administrator/Desktop/CLA/RES/Python/MyoPcaExperiment.py
Number of features: 56
Number of principal components remained: 25
Variance remained: 0.9078136559478833
Contributions for different kinds of features:
MV: 0.8044367686533271  SD: 0.954615964085454  WL: 0.9562466860007007  ZC: 0.9058070450230885  SSC: 0.8743378502966651  AC1: 0.806171425345173  AC2: 0.86457788511082

Process finished with exit code 0
```

Figure 7: This picture shows the results about the principal components.



```
C:\Users\Administrator\AppData\Local\Programs\Python\Python36\python.exe C:/Users/Administrator/Desktop/CLA/RES/Python/MyoSvmTraining.py
Data size: 27832
Precision on training set: 1.0
Precision on test set: 1.0
Precisions for different gestures: {'rest': 1.0, 'up': 1.0, 'down': 1.0, 'left': 1.0, 'right': 1.0, 'open': 1.0, 'close': 1.0}

Process finished with exit code 0
```

Figure 8: This is the training result of the SVM classifier.

4 Conclusion and planning

4.1 Conclusion

A basic system including a SVM classifier was built for solving the static gesture recognition problem, but it is still not good enough to deal with the dynamic gesture recognition problem which not only makes data acquisition a much more harder problem, but also requires a more complex classifier.

4.2 Planning

In the next stage, the dynamic gesture recognition problem will be studied, and before that, an evaluation system like that in [3] should be built to assess a classifier. The problems about the dynamic task will then be focused on:

1. Since dynamic gestures have durations, a mass of data can't be collected in short time;
2. The classifier should make prediction based on the features in a period, which actually increases the number of features greatly;
3. Different dynamic gestures have different durations so the classifier should recognize the beginning and the ending of a gesture;
4. Even the same gesture may have different durations which can also cause many troubles.

Classifiers like RNN can deal with the classification problem on the timeline, so how to use the RNN especially the LSTM in this classification problem will be the main study object.

References

[1] Roberto Merletti and Dario Farina. *Surface electromyography: physiology, engineering, and applications*. John Wiley & Sons, 2016.

[2] Maria Hakonen, Harri Piitulainen, and Arto Visala. Current state of digital signal processing in myoelectric interfaces and related applications. *Biomedical Signal Processing and Control*, 18:334–359, 2015.

[3] Konstantin Akhmadeev, Elena Rampone, Tianyi Yu, Yannick Aoustin, and Eric Le Carpentier. A testing system for a real-time gesture classification using surface emg. *IFAC-PapersOnLine*, 50(1):11498–11503, 2017.