Project #2 Description
Feature Subset Selection

CSE583/EE552 Pattern Recognition and Machine Learning, Spring 2023

Release Date: Tuesday, January 31, 2023
Submission Due Date: Tuesday, February 14, 2023

Contents

1 Overview

1.1 Introduction

This project is about feature subset selection for classification on a human motion (Taiji) sequence dataset. TAIJI is a form of Chinese martial art practiced by millions of people all over the world for its defense training and health benefits. Simplified 24-form TAIJI contains 24 scripted motion sequences. A distinctive feature of TAIJI is its slow and seamless transition, making it difficult to identify the 24 key forms (Figure 1). Furthermore, these 24 key forms can be broken down further into a grand total of 45 key subforms.

The goal of this project is to build classification models to recognize these TAIJI key subforms using feature subset selection methods.

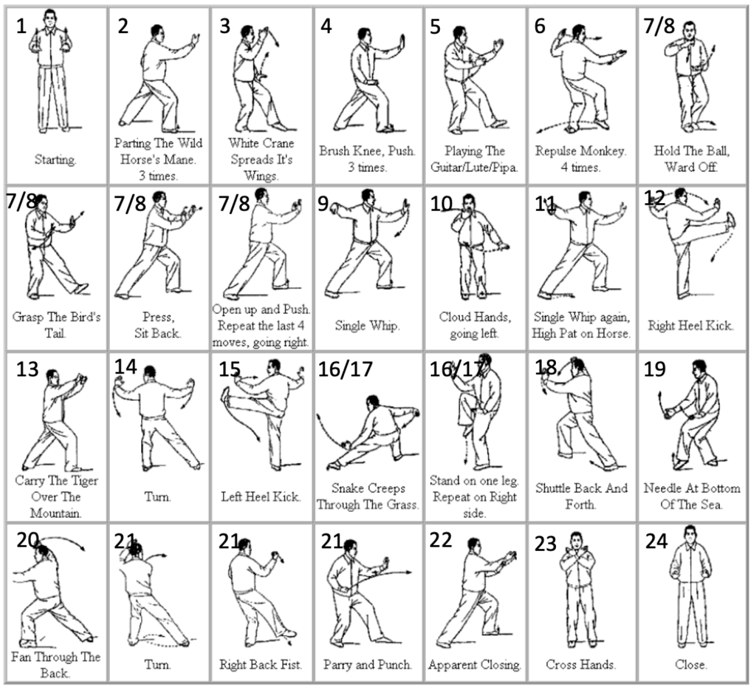
1 Starting.
2 Parting The Wild Horse's Mane. 3 times.
3 White Crane Spreads It's Wings.
4 Brush Knee, Push. 3 times.
5 Playing The Guitar/Lute/Pipa.
6 Repulse Monkey. 4 times.
7/8 Hold The Ball, Ward Off.
7/8 Grasp The Bird's Tail.
7/8 Press, Sit Back.
7/8 Open up and Push. Repeat the last 4 moves, going right.
9 Single Whip.
10 Cloud Hands, going left.
11 Single Whip again, High Pat on Horse.
12 Right Heel Kick.
13 Carry The Tiger Over The Mountain.
14 Turn.
15 Left Heel Kick.
16/17 Snake Creeps Through The Grass.
16/17 Stand on one leg. Repeat on Right side.
18 Shuttle Back And Forth.
19 Needle At Bottom Of The Sea.
20 Fan Through The Back.
21 Turn.
21 Right Back Fist.
21 Parry and Punch.
22 Apparent Closing.
23 Cross Hands.
24 Close.

Figure 1: The 24-form simplified TAIJI forms

We have seen how to project all the features to a lower dimensional space using Fisher Linear Discriminant Analysis in Project #1. Now we are going to learn how to select a subset of features using filters and wrappers. Two key components of feature subset selection are:

1. An **evaluation function** or a **classifier** to measure the discriminative power of the selected feature subset, and

2. A **search strategy** to go through the feature list and select a subset.

1.2 Datasets

The dataset you will be working with is the PSU-TMM100 dataset. (Link to dataset and related research) You will be working on a subset of 47,717 video frames (data points) with 1,961 features (data dimension). The frames are sub-sampled from 5-minute-long Taiji videos performed by 10 different subjects. There are a total of 87 different performances in this subset. The Taiji data consists of two kinds of features:

- **3D body joints feature** from MoCAP (Motion Capture), capturing 17 joints illustrated in Figure 2 (A). Specifically, they are:

    1. Right Shoulder
    2. Right Elbow
    3. Right Wrist
    4. Left Shoulder
    5. Left Elbow
    6. Left Wrist
    7. Right Hip
    8. Right Knee
    9. Right Ankle
    10. Left Hip
    11. Left Knee
    12. Left Ankle
    13. Pelvis Center
    14. Waist
    15. Top of Neck
    16. Clavicle
    17. Thorax

    Each joint contains three variables, x, y, z, indicating the joint location with respect to the center of the hip (average of joints 7 and 10). Thus the body joint feature set has $17 \times 3 = 51$ dimensions.

- **Foot Pressure data**, as shown in Figure 2 (B), a pair (left and right) of foot pressure images with 60×21 pixels for each foot. You are provided with cleaned-up (mask processed) feature dimensions from $60 \times 21 \times 2 = 2520$ to 1910 dimensions.

The two sets of features above are concatenated together to form a total of 1,961 features for each observation (a single data point).

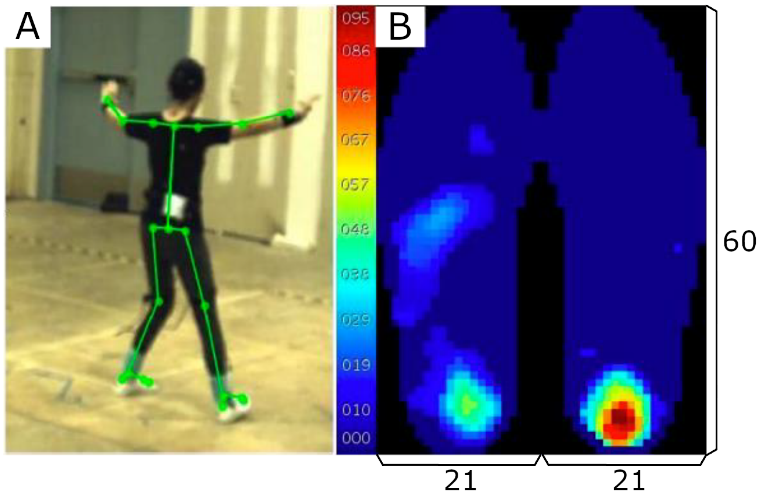

Figure 2: (A): Video data with 17 joints. (B): The corresponding measured left and right foot pressure

CLASS LABELS: Each observation is associated with one of the following classes:

- 0 – NON KEY FRAME (frames in-between two KEY forms).
- 1-45 – ONE OF THE KEY FORMS (names of key forms included in the dataset)

So there are a total of 46 classes.

1.3 Frame labeling strategy

The labeled KEY-form just indicates the end of a Taiji form, given the slow movements of Taiji, the nearby body poses are similar to the KEY pose/form. Thus, we use a range of KEY-form class labels to include those body poses before and after the KEY-forms. For experiments in this project, We use $M = 100$ frames before and $N = 20$ frames after each KEY-form. Also note, we use the class label "0" for that body poses (data points) between two sets of KEY-forms.

1.4 Data splitting strategy

As mentioned in Project 1, we will be using Leave One Subject Out cross validation during classification. That is, for a given subject, all training data will be composed of the other subjects' performances (data), while the testing data will only consist of the selected subject. This allows for the classifier to test on pseudo-unseen data and hopefully more closely resembles a real setting. The provided code will create such data splits for you.

1.5 Implementation

You need to implement a filter and a wrapper feature selection module. For the filter, you need to determine the selection criteria and a ranking subroutine. For the wrapper, you need to choose a classifier for training and testing. You should use one of the built-in classifiers in Matlab or Python library.

More specifically, you will be implementing the following:

1. **Filter Method**
   - This requires a **selection criterion** which you can define yourself, though we highly recommend using the Variance Ratio (VR), Augmented Variance Ratio (AVR) or Minimum Redundancy Maximum Relevance (mRMR) where a higher value indicates higher discriminative power of the feature (for a formal definition, please refer to Section 4.3 in this reference link)
   - You will be ranking the features by sorting their feature criterion values
   - We provide you with code to reduce your feature space to the top K features, but you need to specify the value of K. We recommend selecting the top 10% or top 100 features.
2. **Wrapper Method - Sequential Forward Selection**
   - You will be implementing a sequential forward selection function that takes the top K features from your filter method as input.
   - You choose your own criterion to optimize. (Examples: maximizing classification rate, minimizing false positives, etc.)
3. You will be evaluating the performance over 10 LOSO iterations (different subject left out each time). You need to report the average classification rates and standard deviation of classification rates for both training and testing.

Note: It is not recommended to use more than 100 features when implementing the wrapper SFS algorithm, as it may not be optimal for large feature spaces. We suggest starting with a smaller number of features, such as 15-25, and then gradually increasing the number of features (top 10% or top 100) once the algorithm has been debugged and optimized.

1.6 Starter Code

To get you started, we provide a sample starter code with Taiji data for feature selection, which contains the following:

- Code to load the training and testing data.
- Code to split the data.
- Code to normailize the data.
- Code to visualize the result.

You have all freedom to edit the provided functions (e.g. the visualization code) as you see fit.

Download the starter code with Taiji data from CANVAS.

1.7 Evaluation

For evaluation, you need to evaluate the performance for 10 iterations with each iteration leaving a different subject out. We provide you with a framework for doing LOSO training and evaluation. You need to show classification results (confusion matrix + overall accuracy & std + each class classification rate) averaged on all iterations. You will be reporting this for the following experimental setups:

1. Baseline (train and test a classifier on all features, that is to say without using a filter or a wrapper)
2. Filter only (train and test a classifier on the top K features chosen by your classifier, skip the wrapper step)
3. Filter + Wrapper (train and test a classifier on the subset of features selected by your wrapper that takes the top K features from your filter)

1.8 A Question to be answered

During the lecture, we've introduced dataset size sufficiency issues. Justify that this Taiji data set is or is not sufficiently large for this classification task.

Note: this is an open question. Some references are listed below (feel free to use ANY references you can find as a justification):

- Clinical trial perspective:
  `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2876926/`
- Machine learning perspective:
  `https://machinelearningmastery.com/statistical-power-and-power-analysis-in-python/`

2 Requirements & Grading Criteria

2.1 Requirements

Your report MUST include:

1. The criteria you are using in both your filter and your wrapper.
2. A description or flowchart of the pipeline you used to achieve end classification rates. Note that most of the pipeline is provided in the starter code, but you should still show these steps. Put an emphasis on how your filter and wrapper work.

3. A Histogram of the most discriminative features (top 20 or 30 features).
4. Classification results (confusion matrix + overall accuracy & std + each class classification rate) for Taiji dataset averaged on all iterations for the following configurations.
   - No filter + no wrapper
   - Filter + no wrapper
   - Filter + Wrapper
5. Your own justification to the dataset size sufficiency question (Sec. 1.8)

Make sure to properly cite ALL resources you used for this project.

2.2 Grading Criteria

1. Implementation of Filter (**20 points**)
   (a) Selection criterion. (5 points)
   (b) A scoring and ranking subroutine. (10 points)
   (c) Code implementation. (5 points)
2. Implementation of Wrapper (**40 points**)
   (a) Implement a sequential forward/backward selection algorithm. (10 points)
   (b) Choose your own criterion to optimize. (10 points)
   (c) Choose a classifier for training and testing. (10 points)
   (d) Code implementation. (10 points)
3. Report (**40 points**)
   (a) Flowchart or a description section to show what steps you have done to achieve the final classification results. (5 points)
   (b) A Histogram of the most discriminative features (filter; 20 or 30 top features). (10 points)
   (c) A Histogram of the most commonly selected features (wrapper; 20 or 30 top features). (10 points)
   (d) Classification results (confusion matrix + overall accuracy & std + each class classification rate) for Taiji dataset averaged on all iterations. (10 points)
   (e) Your own justification for the dataset size sufficiency question. (5 points)
4. **Extra Credits (Up to 30 points)**

(a) Better visualization of most discriminative features. For 3D body joints feature, you may highlight the chosen joint in a joint set. For foot pressure feature, you may plot two 3D histogram which x- and y-axis are the length and width of foot pressure map, and z-axis indicates the number of feature been selected by the filter and wrapper among 10 iterations. (Up to 10 points)

(b) Try multiple classifiers, using ensemble learning and other strategies to better classification results. (Up to 10 points)

(c) Compare results between the class balanced data (the main data given for the project) and non class balanced data. The non class balanced data will be released later. (Up to 5 points)

(d) Try different values of M and N for frame labeling and visualize your results. Dataset and starter code for this part will be released later. (Up to 5 points)

3 Submission

Your submission should include the following items:

- Your code, which should be **reasonably commented** on and written in an understandable manner.
- Your written report, which should be following the requirements & criteria listed above.
- A ReadMe document that explains the function of each code file and which data file is used.

Please package all of these items into a single zip file and name it as

FirstName_LastName_ProjectNo.zip.

For example, **John_Doe_2.zip**.

Make sure to properly cite all resources you used for this project and double check your submission before uploading it to the Canvas dropbox.

Please note that graders will read and run your code, so please ensure that your code runs correctly and is well-organized and easy to understand.

4 Common Issues

1. We only be usking numpy, matplotlib, and sklearn classifiers. You may not use any other python library or functions in sklearn other than their classifiers. You may use additionally matplotlib functions or modify the visualiazation code in either Matlab or Python.

2. Providing some built-in classifiers and/or utilities in Matlab and sklearn will be very helpful.

   For the In-built classifiers you can use

   (a) *fitcdiscr, LinearDiscriminatClassifer*
   It creates a linear discriminant classifier, and allows you to classify the data set. use it in conjunction with predict function.
   Tutorial for matlab fitcdiscr and skearn LinearDiscriminantAnalysis.
   For more information about discriminant analysis: `https://www.mathworks.com/help/stats/classification-discriminant-analysis.html`

   (b) *fitctree, DecisionTreeClassifier*
   It creates a binary decision tree for multi-class classification.
   Tutorial for matlab fitctree and sklearn DecisionTreeClassifier.
   For more information about classification trees: `https://www.mathworks.com/help/stats/classification-trees.html`

   (c) *fitcknn, KNeighborsClassifier*
   It creates a k-nearest neighbor classifier.
   Tutorial for matlab fitcknn and sklearn KNeighborsClassifier.
   For more information about classification nearest neighbors: `https://www.mathworks.com/help/stats/classification-nearest-neighbors.html`.

   (d) You may find more suitable classifier for binary or multi-class classification problems here:`www.mathworks.com/help/stats/classification.html` and `https://scikit-learn.org/stable/auto_examples/classification/plot_classifier_comparison.html`

   To take mean diff for n class data set with each class in Matlab, you might need to take combinations and for that use the inbuilt function *nchoosek*. MATLAB tutorial for *nchoosek* is available at `https://www.mathworks.com/help/matlab/ref/nchoosek.html`.

   In cases where you run into NaN values and cannot make any further operations on these values, you need to clear these NaN values. It can be done using the command *Variable(isnan(Variable))=0.*

Notice that this section will be updated if more common issues are asked by the students.