Project #1 Description

Linear Regression + Classification

CSE583/EE552 Pattern Recognition and Machine Learning, Spring 2023

Release Date: Thursday, January 12, 2023
Submission Due Date: Friday, January 27, 2023

Contents

1 Part 1: Linear Regression (45 Points)

1.1 Introduction

The first part of this project focuses on linear regression, which serves as an introduction to important concepts covered in the textbook. You will generate noisy observations $(\mathbf{x}, \mathbf{t})$ of $N = 50$ samples and assume Gaussian noise. Using these observations, you will introduce a prior distribution $\mathbf{p}(\mathbf{w}|\alpha)$ over the coefficients and then solve the Bayesian linear regression problem by estimating polynomial coefficients of different order M, which you can decide on your own to see which one fits better. You will be implementing two different approaches:

1. The ML (maximum likelihood) estimator from a probabilistic perspective. (refer to Equation 1.62, page 29).

2. The MAP (maximum a posteriori) estimator of the Bayesian approach (refer to Equation 1.67, page 30 and Equation 3.55, page 153, use $\beta = 11.1$ and $\alpha = 0.005$ as shown in textbook).

Once you have completed your program, you will need to compare and summarize the results of these two methods. The primary goal of this part of the project is to familiarize you with the Bayesian modeling framework and provide experience using Matlab or Python.

Additionally:

- Background information for this project can be found in sections 1.1 and 1.2 of the textbook by Bishop.

- The starter code for this project is available for download on the course's Canvas page in both Python and Matlab. It is recommended to download it and check the ReadMe.md file for additional details about the starter code.

1.2 Requirements

Your report for Part 1 MUST include the following elements:

1. Derived equations for the two approaches used in the project.

2. Visualization results for the estimated regression models using $N = 50$ sample points. These should include plots to clearly demonstrate how well the models fit the data. You can take Figure 1.3 (page 6) and Figure 3.8 (page 157) of the textbook as a reference. Note: The plotting function is provided in the starter code.

3. A comprehensive summary and comparison of these two methods.

1.3 Grading Criteria of Part 1

- **Part 1 (45 points)**
  - **Maximum likelihood (15 points)**
    - Show the derived equations (and the deriving of the equations). (5 points)
    - Code implementation. (10 points)
  - **Maximum a posteriori (15 points)**
    - Show the derived equations (and the deriving of the equations). (5 points)
    - Code implementation. (10 points)
  - **Write up the report to summarize, compare and contrast the results (15 points)**
    - Figures and tables with nice visualization and description. (5 points)
    - Comparison with different methods and summary. (10 points)
- **Extra Credit (up to 20 points)**
  - Add additional lambda values to the plot of errors for $\ln \lambda = -18, -15, -13$ (Figure 1.8), and you are welcome to use more lambda values. (Up to 5 points)
  - Vary the order of the polynomial M for a fixed number of sample points ($N = 50$), and generate a table similar to Table 1.1 (page 8) that shows the effect on the model fit. (Up to 5 points)
  - Vary the number of sample points N for a fixed degree of polynomial ($M = 9$) and generate a plot similar to Figure 1.6 (page 9) that illustrates the effect on the model fit. (Up to 5 points)
  - Use your results to investigate whether there is an exponential relationship between M and N, known as the "curse of dimensionality" (Up to 5 points)

2 Part 2: Classification (50 Points)

2.1 Introduction

The second part of this project focuses on classification using real data. You will learn how to apply Fisher's linear discriminant technique, which is used for classifying data, to a real dataset of Taiji poses. This part allows you to apply the techniques learned from the course to real-world datasets, and evaluate performance to draw important insights. The dataset is provided on Canvas.

Your implementation should be as follows:

1. First, implement a function to find the Fisher projection using the training features and labels, as described in Bishop 4.1.4 and 4.1.6. You must implement the Fisher projection by yourself.

2. Then, train a classifier using the Fisher projected training data. You can choose from a linear discriminant (Bishop 4.1.1 - 4.1.3), a KNN classifier (Bishop 2.5.2), from Decision Theory using an optimum threshold (end of Bishop 4.1.4), or any other classifier of your choice. You can use either the built-in functions of Matlab or implement them by yourself. For Python, you again may optionally implement the classifier yourself or import one from a designated library (details will be included in the readme). This function should return the Fisher projection coefficients and the corresponding fitted classifier necessary for the testing function.

3. Finally, test the performance of your projection and classification method on the datasets given by either built-in functions or by implementing them by yourself. Quantitatively evaluate the classification method on the given dataset.

2.2 About the Taiji Pose Dataset

This is a dataset of the joint angles (in quaternions) of 35 sequences from 5 people performing Taiji in our motion capture lab. The dataset contains 8 classes consisting of 7 different Taiji poses, as well as transitional poses (the '0' class). Here is a sample video of one of the performances: link to the video (We are only using up to 1:30 in the video). The data we will be working with has $N = 15285$ observations and $D = 64$ dimensions.

2.3 Training

Note we will be 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.

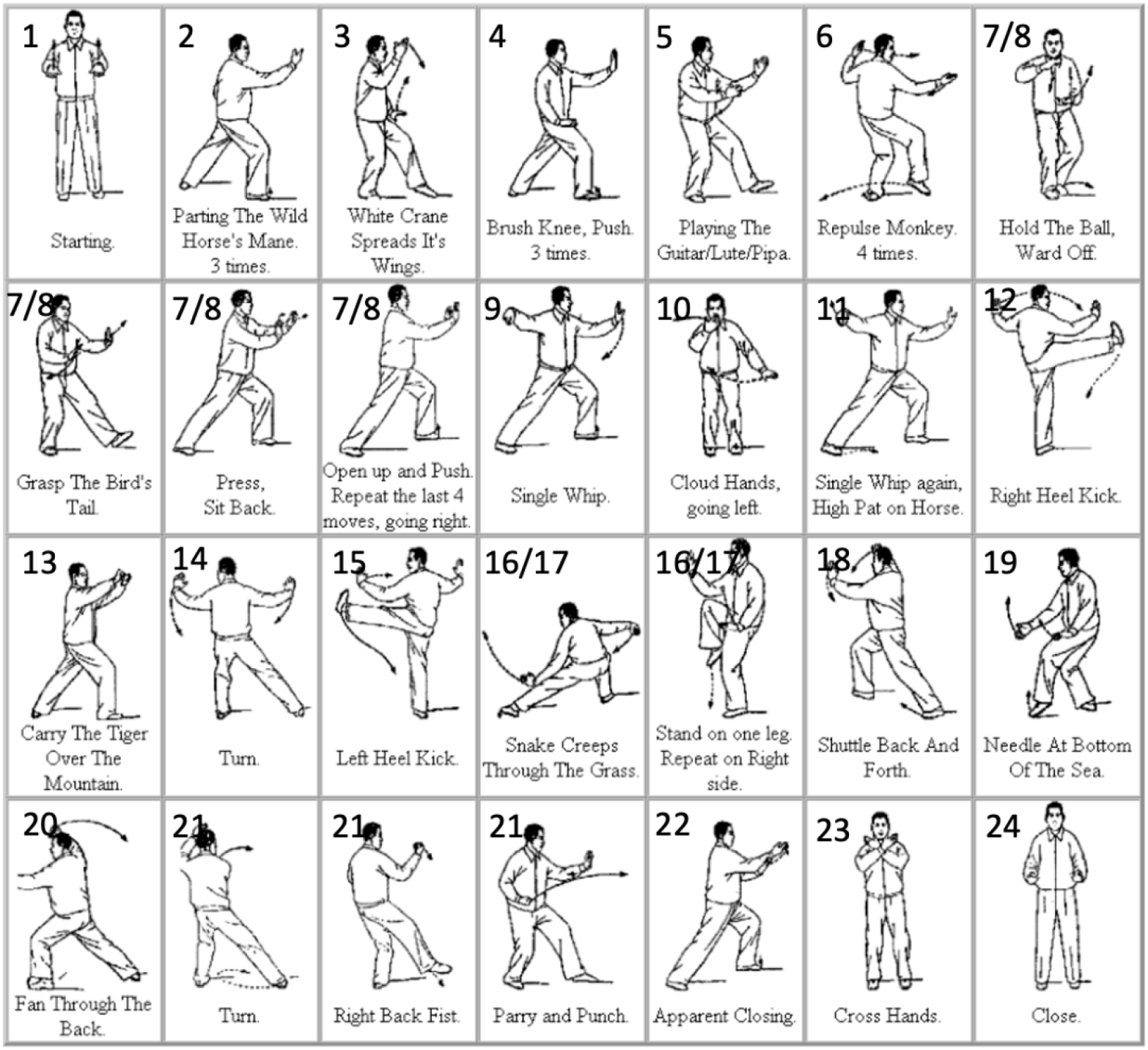

Figure 1: The 24-form simplified TAIJI forms. Note the data we will be using is a slightly simplified version which contains fewer classes.

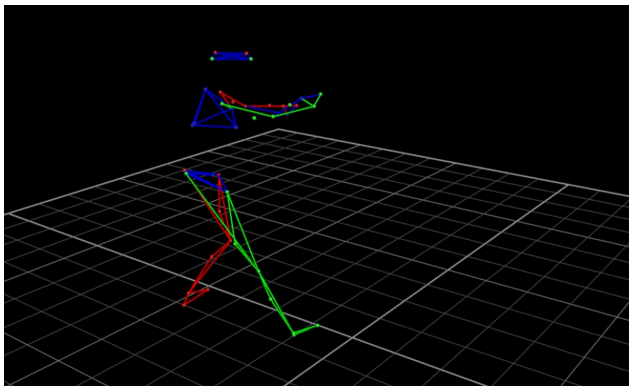

Figure 2: One snapshot of a MoCAP frame of the Taiji Pose.

2.4 Requirements

Your report for Part 2 must include:

1. A clear presentation of the equations that define your Fisher projection and the estimated model parameters.
2. Classification results on both the training and testing data. This must include:
   (a) Confusion matrices. Note: The function for creating confusion matrices is provided in the starter code.
   (b) The classification rates for each class and an overall classification rate.
3. A thorough analysis of your results in the report, not just a collection of figures and tables. Explain the meaningful observations behind the numbers and compare the classification results on training and testing data. Explain any overfitting problem if found.
4. Identification and explanation of any outliers in the data, and how they affect the classification results. If no outliers are found, provide an explanation of this point with data visualization.

Make sure that the report is not just a series of tables and figures, but a thoughtful analysis of your results that is explained in a clear and concise manner.

2.5 Grading Criteria of Part 2

- Part 2 (**50 Points**)
  - Implementation of Fisher projection (**20 points**)
    - Provide the closed-form weight equation (providing the derivation steps will also be sufficient) (5 points).

    - Code implementation (15 points)
  - Use/Implement a classifier to classify the data after the Fisher projection (**10 points**)
  - Results on the dataset (**20 points**)
    - Confusion matrix (5 points)
    - Classification rates for each class and overall classification rate (5 points)
    - Analysis and conclusion (10 points)

- **Extra Credits (Up to 15 points)**
  - Demonstrate how Fisher criterion is a special case of least squares (Bishop 4.1.5) using samples from any two classes of a dataset. (5 points)
  - Visualize the decision boundaries by projecting the features to 2 dimensions. Note: The function of drawing decision boundaries is provided in the starter code. (5 points)
  - Plot the training/testing data points, indicating the wrongly classified points based on your results from the confusion matrix for Fisher projection + KNN or other classifiers you use, by modifying the visualize function. (5 points)

3 Part 3: Question about Central Limit Theorem (5 points)

You are required to answer the following question in a separate section at the end of your report.

1. Explain what the "central limit theorem" is and provide references to where the concept is covered in your learning materials.

2. For a coin with a probability of heads of 0.6 in each flip, calculate the exact distribution of the total number of heads when flipped 5 times. Provide the equation(s) used to generate the result and a brief explanation of your reasoning.

3. Using the central limit theorem, approximate the distribution of the total number of heads when the coin is flipped 5000 times. Provide the equation(s) used to generate the result and a brief explanation of your reasoning.

Make sure to clearly explain your thought process and show your work for both parts of this question.

4 Submission

Your submission should include the following items:

- Your code, which should be **reasonably commented** on and written in an understandable manner.
- Data files that you used to estimate each of the regression models.
- 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 to generate which model.

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

FirstName_LastName_ProjectNo.zip.

For example, **John_Doe_1.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.

5 Common Issues

Familiarize yourself with Matlab by reviewing the online tutorials and Matlab help documents. For those using Python, we will make use of Numpy, a package for scientific computing to work with arrays.
Don't try to do this at the last minute.
Good luck!

1. I run into error when running the Matlab starter code. How to fix it?
   Kindly check if Image Processing Toolbox has been installed to your Matlab.

2. How to solve the linear system of equations?
   You can use the backslash operator "\" rather than the inverse function to solve the linear system of equations in Matlab. Kindly check `https://www.mathworks.com/help/matlab/ref/mldivide.html` for more reference. For Python users, Numpy provides support for linear algebra computations with numpy.linalg.

3. How to compute eigenvalue & eigenvector?
   You can use Matlab built-in function eig for computation. Check numpy's linear algebera support linalg.

4. How do I install/import numpy and other libraries used in the python code?
   We highly suggest you use a tool for creating environments to manage python version(s) and packages. We recommend Anaconda for this and numpy's website provides a tutorial. Package installation can be achieved with *pip*. The Python starter code's readme will further detail package use.

5. How to convert training label vector of length N into a $N \times K$ matrix?
   The idea is to use 1-of-K coding scheme, which is to use a vector $\mathbf{t}$ of length K to represent a label with K classes. Such that if the class is C_j, then all elements of $\mathbf{t}$ are set to zero except t_j set to one.

   For example, we have $K = 5$ classes. And the label is class 3, then its corresponding 1-of-K coding will be:

   $$\mathbf{t} = (\mathbf{0}, \mathbf{0}, \mathbf{1}, \mathbf{0}, \mathbf{0})^{\mathbf{T}}$$

   And finally, for training label vector of length N, we can use this scheme to expend it to a $N \times K$ matrix.

6. How to do classification by Fisher Projection?
   Notice that fisher projection can ONLY do dimension reduction but not classification. The goal of fisher's linear discriminant is to find a projection that maximized the class separation. Hence, after implementing Fisher Projection, you are required to train a

classifier to the projected training data (in a reduced dimension). You are welcome to use any classifier from lecture or based on your own knowledge.

7. How do I generate plots?
   You may use the provided plotting/confusion matrix functions in the starter code. The starter code contains example showing the functions creating a plot, but you will likely want to change the data being plotted.

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