Algorithms and software for the analysis of massive genetics datasets

An independent study by,

Kiran Gilvaz

Sapan Shah

Under the guidance of,

Prof. Petros Drineas

Contents

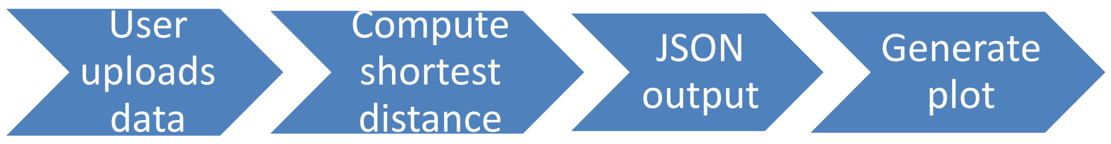
1. Introduction

1.1 Overview

This document details the software documentation for a product that was designed to estimate population ancestry using statistical algorithms on massive genetic datasets. The product was designed to accept genetic data from the user via a web browser. The application computes the users nearest neighbors and sends out an email with the generated results and a link to an interactive 3D plot.

1.2 Architecture

Python was selected as the primary programming language due to the significance it has gained in computer science over the years and the numerous features it supports with great ease. The web application was developed using Django; a popular open source web framework. The program accepted the user data via a web application, computed the results and generated data in JSON format to decrease cross application dependency. This JSON data was further fed as input to a third party library called Canvas Express to create an interactive 3D interface. The functional framework of the application can be summarized by the following figure.



1.3 Input / Output

The user is prompted to enter a valid email address and uploads a (643862 x 1) CSV file. Once the data has been processed, the user receives an email with 2 static 2D and 3D plots as attachment, top 20 closest regions and a link to an interactive 3D plot.

2. Requirements and Installation

2.1 Software requirements

Programming language: Python 2.7

Web framework: Django 1.4

Web server: Apache 2.2

Libraries used:

- Canvas Express: Javascript library used to generate interactive 3D plots
- Python-Matplotlib: Python plotting library
- Python-scipy: Python library for MatLab I/O operations
- Python-numpy: Python library for complex mathematical operations
- Python-json: Python library to serialize python objects to JSON type
- Python-smtplib: Python library to send emails

2.2 Hardware requirements

Due to the programs excessive computational needs, we recommend a server with these minimum requirements.

- Processor	: Intel 2.4GHz (Single Core)
- Memory	: 1GB

However, the program was tested on a system with the following specifications.

- Processor	: Intel i7 Extreme (8 Cores)
- Memory	: 8GB

A system with the above specifications would compute the results in approximately 1 hour.

2.3 Installation

The following describes the installation procedure on a Windows platform. The operating system is assumed to be Windows 7.

1. **Python**

   http://www.python.org/download/releases/2.7.3/

2. **Scipy & Numpy**

   The easiest recommended Windows installation would be to download the ‘super pack’ binary distribution of Scipy and Numpy from,

   http://sourceforge.net/projects/numpy/files/NumPy/

   http://sourceforge.net/projects/scipy/files/scipy/

3. **Matplotlib**

   https://github.com/matplotlib/matplotlib/downloads

4. **Download project files**

   Project files can be downloaded from the repository at

   https://github.com/kirangilvaz/gene-data

5. **Setting up the sever**

   The following steps are described in detail at

   http://www.venkysblog.com/setup-django-on-windows-with-apache-and-mysql

   Django uses the mod-wsgi adapter to connect to Apache’s server. To manage our configuration easily, this documentation uses Wamp Server as our ‘Windows Web development environment’.Wamp Server is not required is the application is deployed on a Linux based server. A wrapper installation package that includes Mod-Wsgi and Apache can be downloaded from

   http://www.wampserver.com/en/#download-wrapper

   Download the appropriate mod_wsgi file from

   http://code.google.com/p/modwsgi/wiki/DownloadTheSoftware?tm=2

   and rename the file to “mod_wsgi.so” (done as sanity check)

   Once downloaded and installed, the server is configured with the following steps,

   Add these lines to the httpd.conf file

   ```
   LoadModule wsgi_module "<Location of mod_wsgi.so file>"
   WSGIScriptAlias "/" "<location of django.wsgi, present inside project path>"
   <Directory "location of project directory geneticanalysis/">
   Order allow,deny
   Allow from all </Directory>
   ```

   They add the mod_wsgi plugin to apache, alias for the context path and directory permissions.

6. Configuring Django

geneticanalysis/settings.py

Ensure that the following parameter is set,

TEMPLATE_DIRS points to “/gene/templates/”

This is the location of the upload.html file.

gene/gene.py

geneDataDirectory points to the location of the “*.mat” files. This folder must contain all the necessary .mat files including ‘SNchr1A.mat’ through ‘SNchr22A.mat’, ‘SNchr1B.mat’ through ‘SNchr22B.mat’, along with ‘regions.mat’ and ‘SET1.mat’. Please note that the file names are case sensitive. Altering names would require you to change them inside the ‘gene.py’ file in the project directory.

outputDirectory points to the location where you want the JSON, static plots to get saved. Please check that this directory has write permissions and can be directly accessed via a public URL. It is recommended that you place your plotting HTML code in this URL. For example:

http://www.geneticanalysis.com/generateplot.html?data=filename.json

In the above example, ‘generateplot.html’ and the generated json file are in the same directory that can be directly accessed via the URL.

downloadDirectory points to the location where the user uploaded file gets saved.

gene/utility.py

outputDirectory and *downloadDirectory* must hold the same values as above.

plotUrl points to the URL of the html page that generates the 3D interactive plot. The program assumes that the ‘html page’ accepts the JSON file name in the parameter ‘data’. So the program appends “?data=<filename.json>” to this URL.

fromEmail, smtpServer, smtpUsername,smptPassword and smtpPort also need to be set to ensure email delivery.

/web

This folder contains the HTML code to generate the 3D interactive plot. Place this file in the path specified in the *outputDirectory* variable mentioned above.

3. System design

3.1 Django

A detailed description of how to setup Django and its various parts can be found at https://docs.djangoproject.com/en/dev/intro/tutorial01/. However, for the purposes of this documentation we will describe parts relevant to this project.
/geneticanalysis is the project root folder.
/geneticanalysis/geneticanalysis/settings.py contains project configuration data.
/geneticanalysis/geneticanalysis/urls.py is used to define the URLs and associate them to their respective ‘views’.
/geneticanalysis/gene/ contains all the files necessary for the application.

3.2 Component description

3.2.1 urls.py

This file is used to associate the URL to its ‘view’.
In our program we set 2 URLs. **/gene/upload** and **/gene/fileupload** is associated to **upload()** and **FileUpload()** functions inside view.py respectively.

3.2.2 views.py

The **/gene/upload** uses Django templates to generate an HTML page upload.html that provides the user with an interface to upload data and enter an email address.
The template **upload.html** located under **/geneticanalysis/gene/templates/.**
It contains a function called **validate()** that is used to check if the user enters a valid email address. It also contains an auto generated “**csrf_token**” is used to check against Cross-site request forgery attacks. If the validation is successful, the form posts the content to **/gene/fileupload** that handles the uploaded data.
FileUpload function in views.py sends the file to utility.py that handles the file on the server.

3.2.3 Utility.py

This file contains 3 functions.

handle_uploaded_file(fileObj, email)

This function accepts 2 parameters. The email address of the user uploading the file and the file object that contains the uploaded file data. This function saves the uploaded file on physical storage in multiple chunks and generates a customized filename for all later data generated. This function further calls ProcessData.

ProcessData(filename,outputFileName,email)

This function accepts 3 parameters. The custom filename that defines all future naming conventions, the filename of the user's data on the physical storage and the email address of the user. This program uses this information to spawn a thread that initializes the computation. Once the process completes, the sendMail function is called to send an email to the user.

sendMail(send_from, send_to, subject, text, fileName)

This function is used to send an email to the user with the generated results.

3.2.4 gene.py

This file can be considered as the heart of the program. A major portion of the computation code lies is contained within this file.

processThis(fileName,userFileName)

This function accepts the custom filename and the filename of the uploaded file. This function performs the following operations.

- Reads the user input vector
- Reads and loops over the number of mat files to performs the following:
  - Filters the columns by data inside 'SET1.mat'
  - Appends data from the user vector
  - Calculates the mean center
  - Perform $T = T+(A^T.A)$
- Calculate U,E2,V= SVD (T)
- Return U

generateOutput(matrix,filename)

The matrix holds the results of the SVD computation and the filename is the custom filename used to save the results. The generateOutput function is used to create a JSON array with 4 fields. ‘x’,’y’,’z’,’region’. They hold the x,y,z coordinates along with the respective regions which they belong to. The coordinates are essentially the values from the first 3 columns from each row in the matrix. The regions are retrieved from the ‘regions.mat’ file. The last row is marked as ‘INDIVIDUAL’ as it contains the coordinates of the individual from the user vector. This function also generates a 2D plot and a 3D plot that is later used as attachments while sending an email to the user.

euclideanDistance(matrix)

This function takes the resultant matrix from the SVD computation and calculates the nearest neighbors for each coordinate. The resultant array is then sorted in ascending order and attached to the email message.

4. Interactive 3D plot

This section details how the JSON output from the Python code is used generate a 3D interactive plot on a web page using a JavaScript library called 'Canvas Express'.
The project home page is located at http://www.canvasxpress.org/.

4.1 Requirements

Web browser: Mozilla Firefox / Chrome (to see output).
It is recommended that you use 'Chrome Frame' while using Internet Explorer.
Library files:

- scripts/canvasutilities.js
- scripts/canvasXpress.min.js
- scripts/excanvas.js

4.2 Installation

Copy the contents of the /web directory into a publically accessible directory within the web server. This includes 'index.html' and 3 JavaScript files under the /script folder. The plot is generated by passing the output JSON file as a parameter to the index.hml file. For example: http://www.geneticanalysis.com/plot/index.html?data=jsonfile.json

4.3 Component description

Libraries included

The following imports the required library files.

<script type="text/javascript" src="script/canvasXpress.min.js"></script>

<script type="text/javascript" src="script/excanvas.js"></script>

<script type="text/javascript" src="script/canvasutilities.js"></script>

<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.8.3/jquery.min.js"></script

getUrlVars()

Is used to extract the value of the parameter used in the URL.
For example: getUrlVars()["data"] retrieves the value for the parameter 'data'.

loadData()

This function is used to make an AJAX call to retrieve the contents of the JSON file into the program. It is called on page load. Once loaded that serialized JSON string is passed as a parameter to the parseJson function.

parseJson(jsonData)

Accepts the JSON string as a parameter and calls the showdemo() that plots the content on a 3D scatter plot. The JSON data is parsed using the jQuery JavaScript library.

showdemo(jsonArray)

The following code generates a Scatter 3D plot using the Canvas Express engine.

new CanvasXpress("canvas", {y: {smps: ["X","Y","Z"],data:jsonArray}},{graphType: "Scatter3D",xAxis: ["X"],yAxis: ["Y"],zAxis:["Z"],scatterType:false,show3DGrid:true})

HTML body

The following creates a canvas to display the 3D scatter plot.

<canvas id=”canvas” width=”1024” height=”900”></canvas>

The legend is also manually set in the web page.

5. Future work

5.1 Scalability

The program was designed to output data in a universal JSON format to facilitate scalability. Hence, more intuitive 3D visualizing tools could be developed using the JSON output from the python code. Internet Explorer users could be provided with a desktop application that can be used to visualize the results.

A 3D visualization tool worth researching is a subset of “Project R”
http://cran.r-project.org/web/packages/scatterplot3d/index.html

5.2 Computing time

Computing time can be improved by optimizing the code or preprocessing the data.