Criterion B-Design

Word Count: 179

The client should be able to calculate the probability of getting admission into a particular choice of college based on the input data (features). The program must first train a neural network- based on a training dataset- to obtain optimized weights for calculating the probability, then testing the optimized weights obtained on a testing dataset and saving the AUC Score Calculated.

Hence, a raw college dataset is obtained from the US Govt Website[1], and stored as a .csv file (This is a sample).

Fig 1: Raw College Dataset

	A	B	C	D	E	F	G
1	SAT Score	ACT Score	High School GPA	IB Predicted Score	In-state/Out-of-State Status	Choice of Major	Admission Status
2	1280	27	3.5	36	Out-of-State	FinanceandAccounting	Accepted
3	1100	22	3.1	32	Out-of-State	History	Rejected
4	1280	27	3	31	In-State	FinanceandAccounting	Accepted
5	1130	23	3.5	36	Out-of-State	Psychology	Accepted
6	1470	33	4	42	In-State	Engineering	Accepted
7	1030	20	3.7	38	In-State	KinesiologyandPhysicalTherapy	Accepted
8	1130	23	2.5	26	Out-of-State	History	Rejected
9	1170	24	2.9	30	Out-of-State	History	Accepted
10	1130	23	3	31	Out-of-State	ForeignLanguages	Rejected
11	1500	34	4	42	Out-of-State	Communications	Accepted
12	820	14	3.3	34	Out-of-State	Psychology	Accepted
13	730	13	2	21	Out-of-State	Art	Rejected
14	1030	20	3.1	32	Out-of-State	HealthProfessions	Accepted

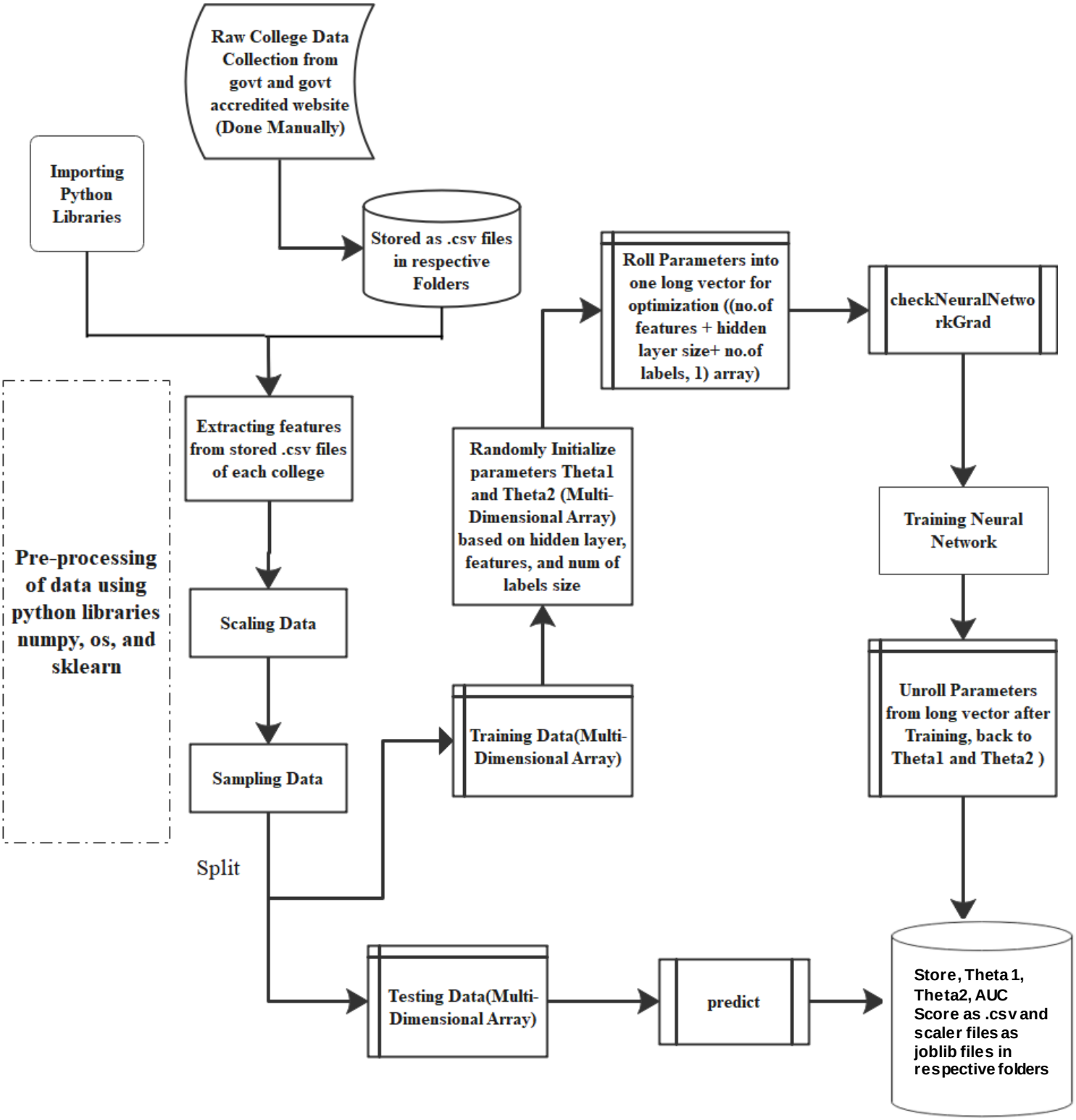
Since a few features mentioned are in the form of a String Datatype, these features are mapped to an integer based on the number of labels.

[1] *Common Data Set*, 18 Nov. 2000, commondataset.org/.

Fig 2: Processed college dataset

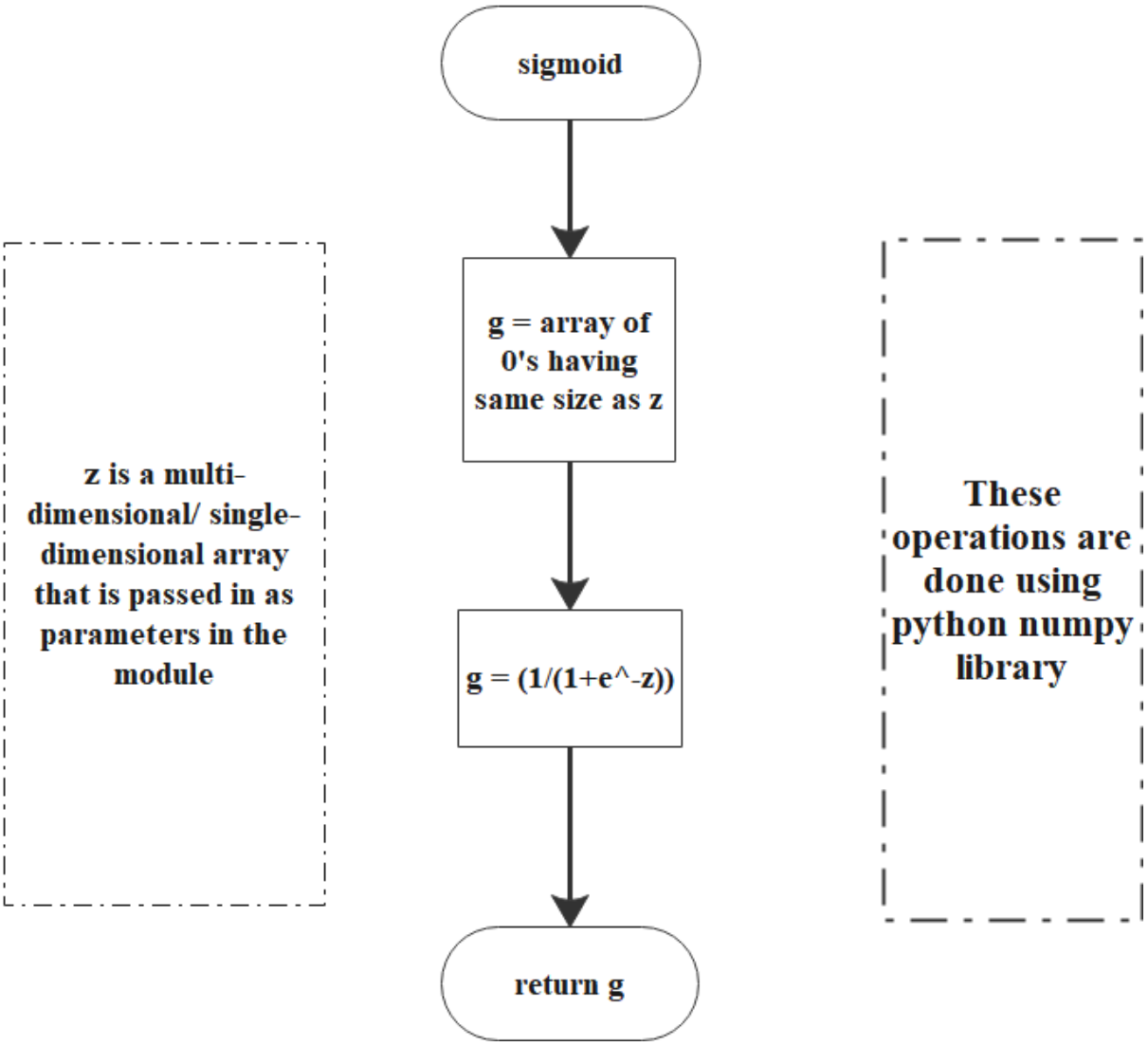
	A	B	C	D	E	F	G
1	SAT Score	ACT Score	High School GPA	IB Predicted Score	In-state/Out-of-State Status	Choice of Major	Admission Status
2	1280	27	3.5	36	10	65	1;
3	1100	22	3.1	32	10	69	0;
4	1280	27	3	31	9	65	1;
5	1130	23	3.5	36	10	79	1;
6	1470	33	4	42	9	61	1;
7	1030	20	3.7	38	9	81	1;
8	1130	23	2.5	26	10	69	0;
9	1170	24	2.9	30	10	69	1;
10	1130	23	3	31	10	66	0;
11	1500	34	4	42	10	56	1;
12	820	14	3.3	34	10	79	1;
13	730	13	2	21	10	52	0;
14	1030	20	3.1	32	10	68	1;

To obtain optimized value of weights for calculating the probability, few tasks are to be completed in a systematic and logical manner. Each of these tasks are programmed as sub-modules, which will then be used by a main python program (integration) to train, optimize and store the weights. Using these weights, a dataset is tested and made available to GUI script that calculates the required probability of a particular college.

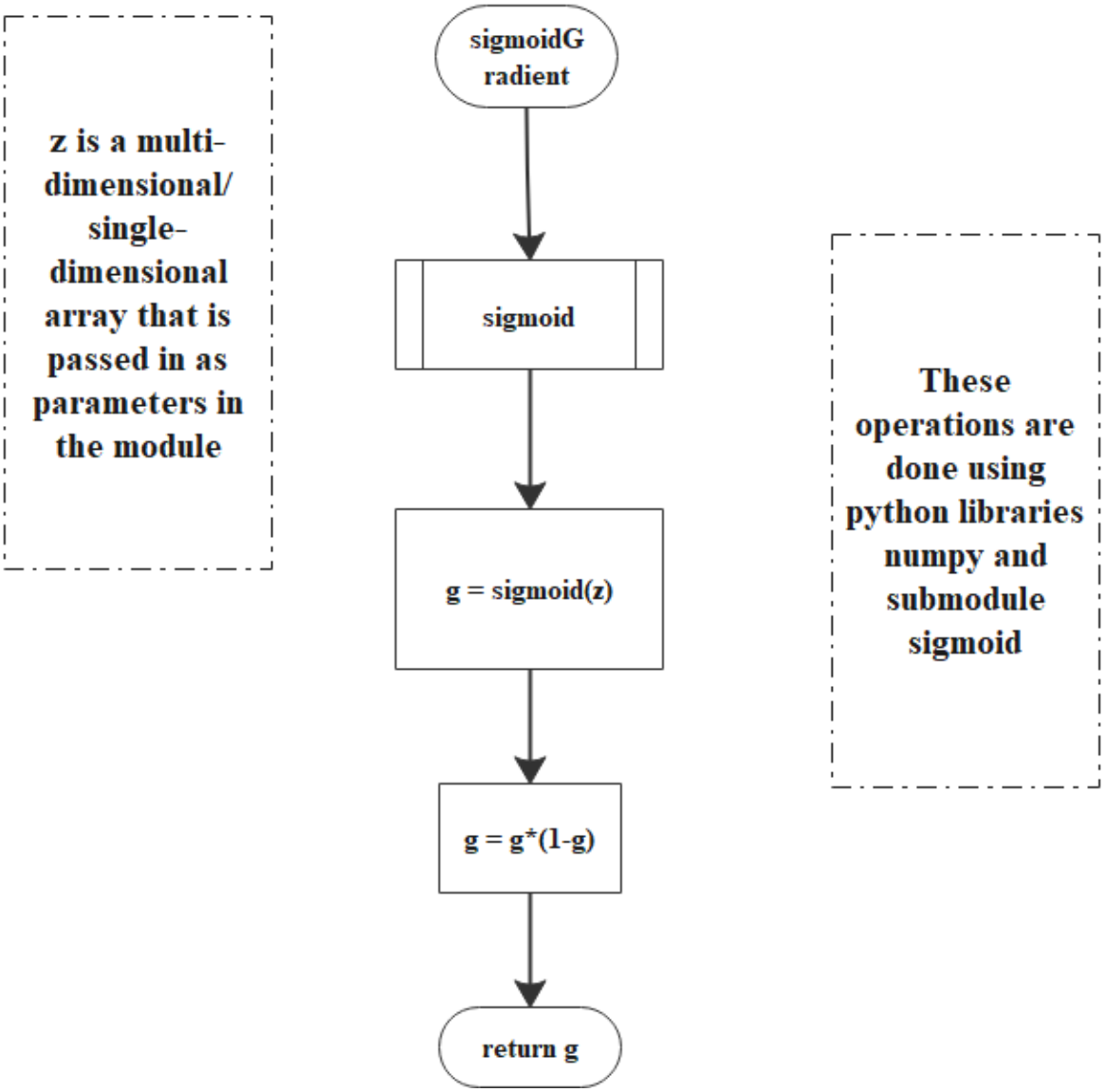

Raw College Data Collection from govt and govt accredited website (Done Manually)
Importing Python Libraries
Stored as .csv files in respective Folders
Roll Parameters into one long vector for optimization ((no.of features + hidden layer size+ no.of labels, 1) array)
checkNeuralNetworkGrad
Extracting features from stored .csv files of each college
Randomly Initialize parameters Theta1 and Theta2 (Multi-Dimensional Array) based on hidden layer, features, and num of labels size
Training Neural Network
Pre-processing of data using python libraries numpy, os, and sklearn
Scaling Data
Unroll Parameters from long vector after Training, back to Theta1 and Theta2)
Sampling Data
Training Data(Multi-Dimensional Array)
Split
Testing Data(Multi-Dimensional Array)
predict
Store, Theta 1, Theta2, AUC Score as .csv and scaler files as joblib files in respective folders

Process Flowcharts (Sub-Modules)

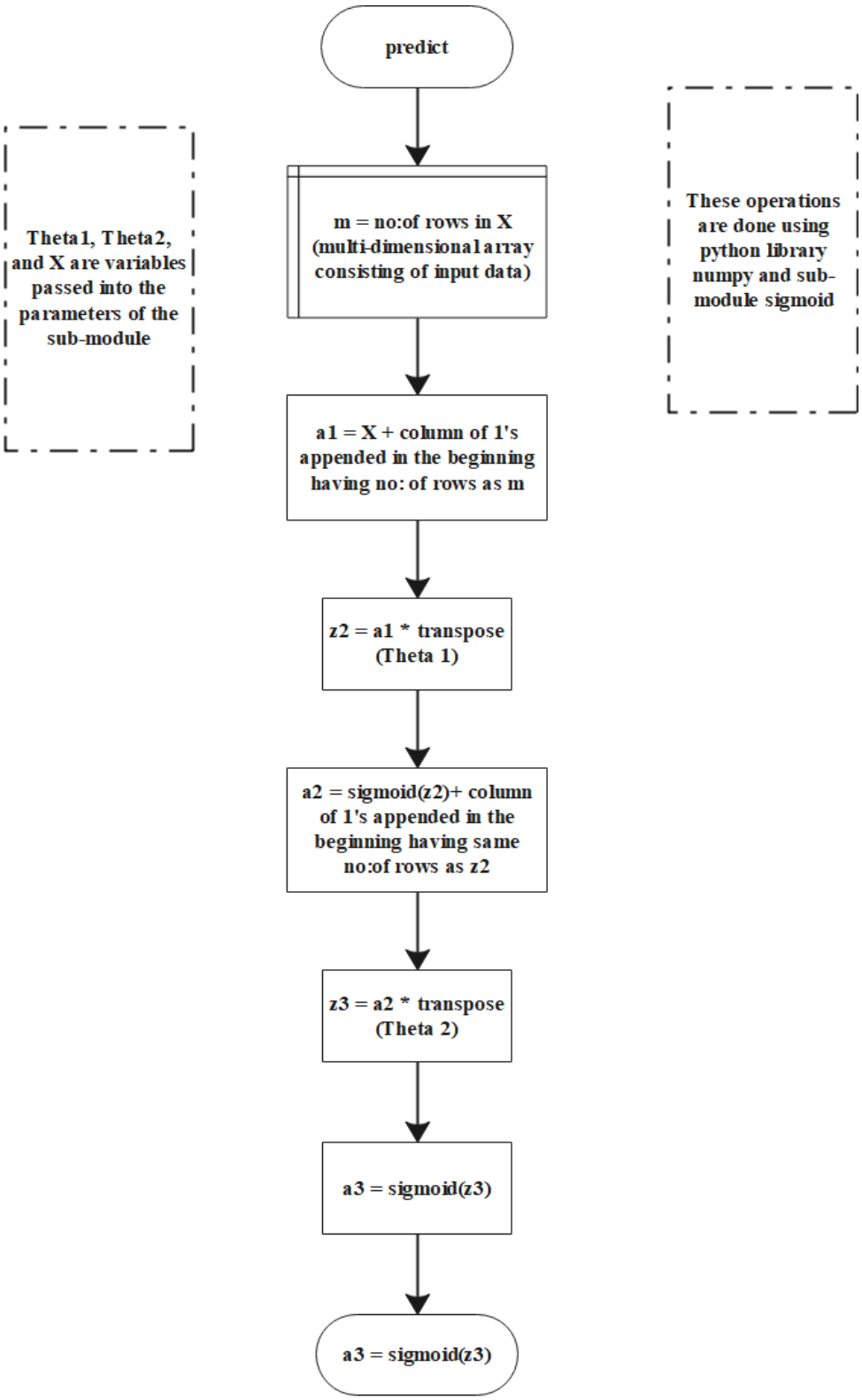
Sigmoid



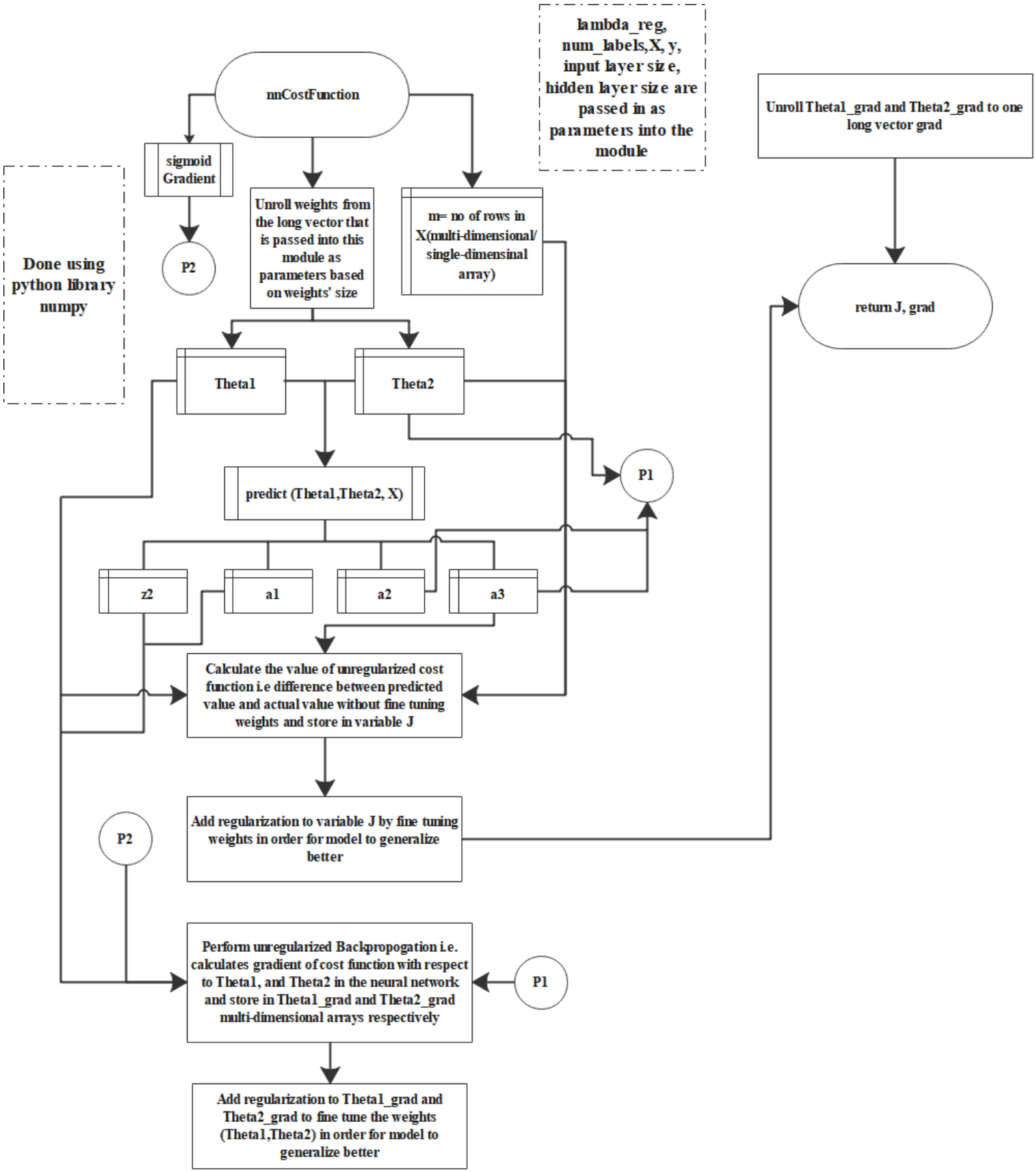
sigmoidGradient



predict


predict
Theta1, Theta2, and X are variables passed into the parameters of the sub-module
m = no:of rows in X (multi-dimensional array consisting of input data)
These operations are done using python library numpy and sub-module sigmoid
a1 = X + column of 1's appended in the beginning having no: of rows as m
z2 = a1 * transpose (Theta 1)
a2 = sigmoid(z2)+ column of 1's appended in the beginning having same no:of rows as z2
z3 = a2 * transpose (Theta 2)
a3 = sigmoid(z3)
a3 = sigmoid(z3)

nnCostFunction



$$J = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log\left(a3^{(i)}\right) - (1-y^{(i)})\log\left(1 - a3^{(i)}\right)\right]$$

$^{(i)}$ *denotes the* i^{th} *sample of column vector. Example,* $y^{(3)}$ *represents the* 3^{rd} *value in column vector* y *and* $a3^{(1)}$ *represent the* 1^{st} *value in column vector* $a3$

Fig 1: Calculation of Unregularized Cost function[2]

$$J = J + \frac{\lambda}{2m}\left[\sum_{j=1}^{hidden\_layer\_size}\sum_{k=1}^{input\_layer\_size}\left(Theta1_{j,k}\right)^2 + \sum_{k=1}^{hidden\_layer\_size}\left(Theta2_{1,k}\right)^2\right]$$

$_{j,k}$ *j refers to the row number and k refers to column number. For example,* $Theta1_{1,2}$ *is the value in row* 1, *column* 2 *of the weight* $Theta1(multi-dimensional\ array)$. $\lambda(lambda)$ *is the regularizing factor*

Fig 2: Adding Regularization to Cost function[3]

$$delta3 = a3 - y$$

$$delta2 = (delta3 * Theta2) .* sigmoidGradient(z2)$$

$$(remove\ bias\ unit\ from\ delta\ 2\ i.e.\ column\ 1)$$

$$Theta2\_grad = (delta3.T * a2) * \frac{1}{m}$$

$$Theta1\_grad = (delta2.T * a1) * \frac{1}{m}$$

$$(sigmoidGradient\ refers\ to\ the\ sub-module'sigmoidGradient')$$

Fig 3: Perform Unregularized Backpropogation[4]

---

[2] NG, Andrew. *Basics of Neural Network Programming*. cs230.stanford.edu/files/C1M2.pdf.

[3] NG, Andrew. *Basics of Neural Network Programming*. cs230.stanford.edu/files/C1M2.pdf.

[4] NG, Andrew. *Basics of Neural Network Programming*. cs230.stanford.edu/files/C1M2.pdf.

$$Theta2\_grad = Theta2\_grad + \frac{\lambda}{m} * (Theta2\_grad)$$

$$(Exclude\ 1st\ column\ during\ calcuation\ as\ it\ has\ bias\ term\ i.e.\ 1)$$

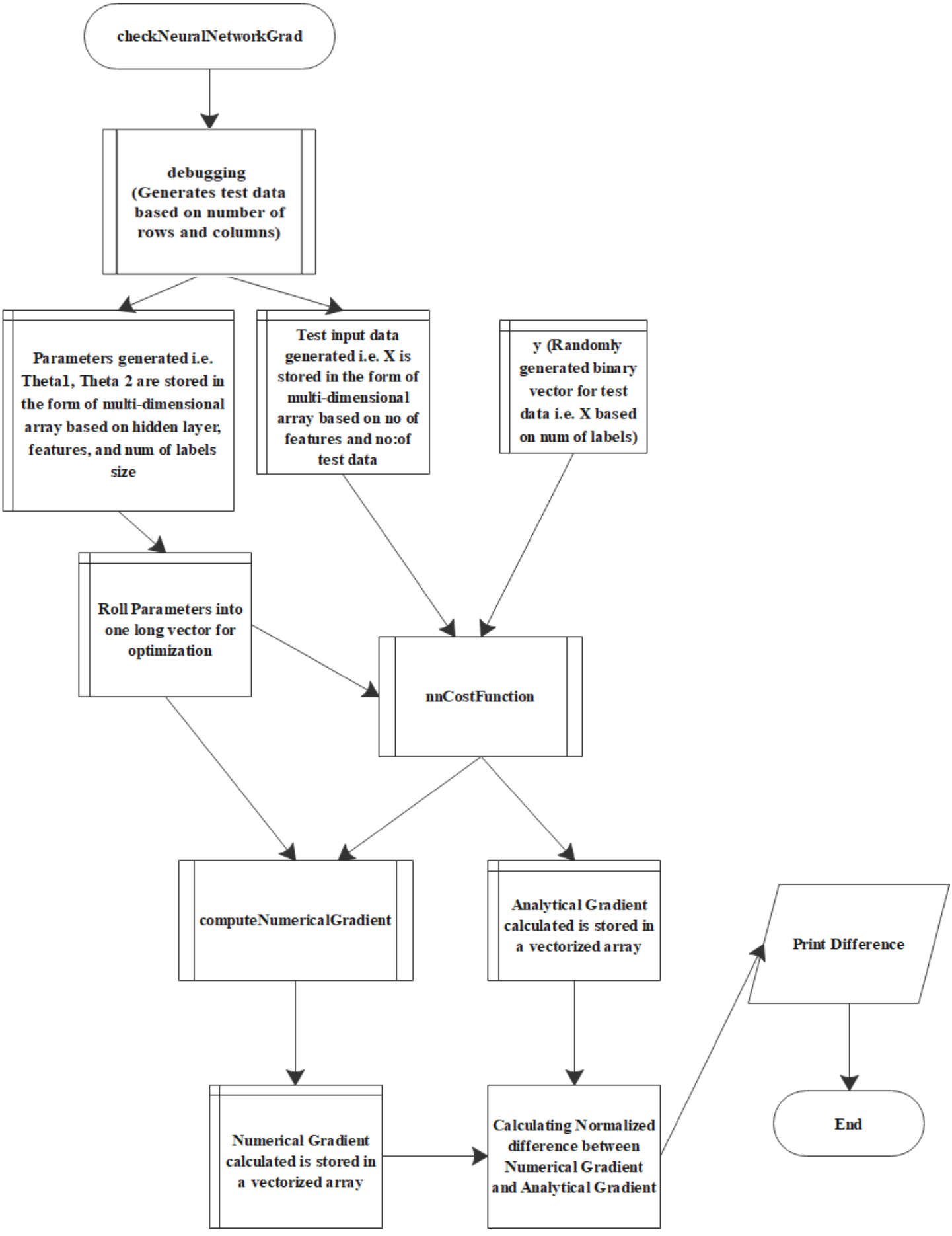
$$Theta1\_grad = Theta1\_grad + \frac{\lambda}{m} * (Theta1\_grad)$$

$$(Exclude\ 1st\ column\ during\ calcuation\ as\ it\ has\ bias\ term\ i.e.\ 1)$$

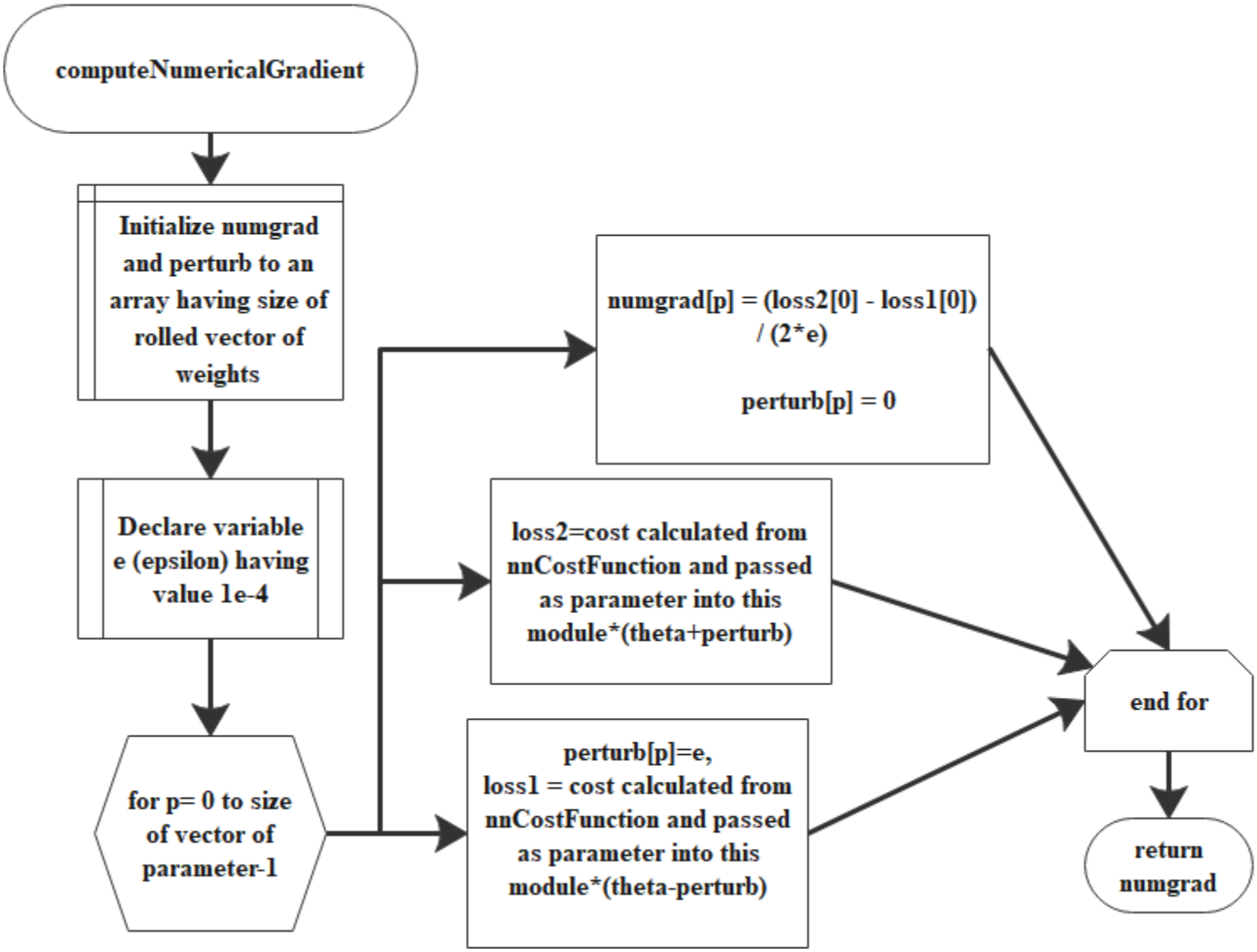
Fig 4: Adding Regularization to Theta1_grad, Theta2_grad[5]

[5] NG, Andrew. *Basics of Neural Network Programming*. cs230.stanford.edu/files/C1M2.pdf.

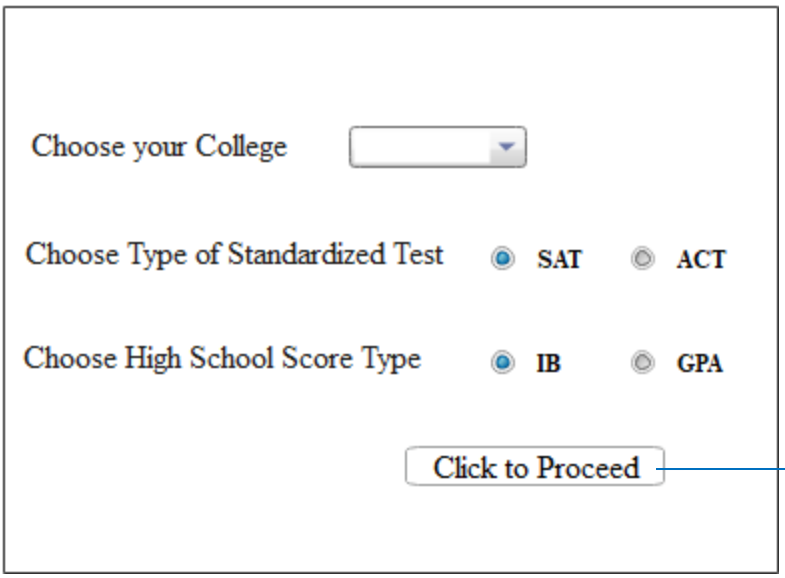
checkNeuralNetworkGrad



computeNumericalGradient



Screen Diagrams for GUI



Main Window

When a Public College is Chosen

Fig 3

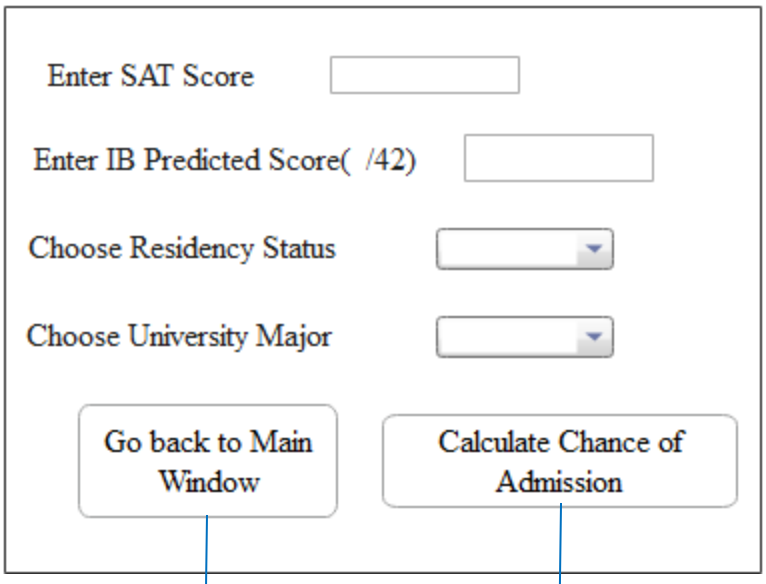


Fig 4

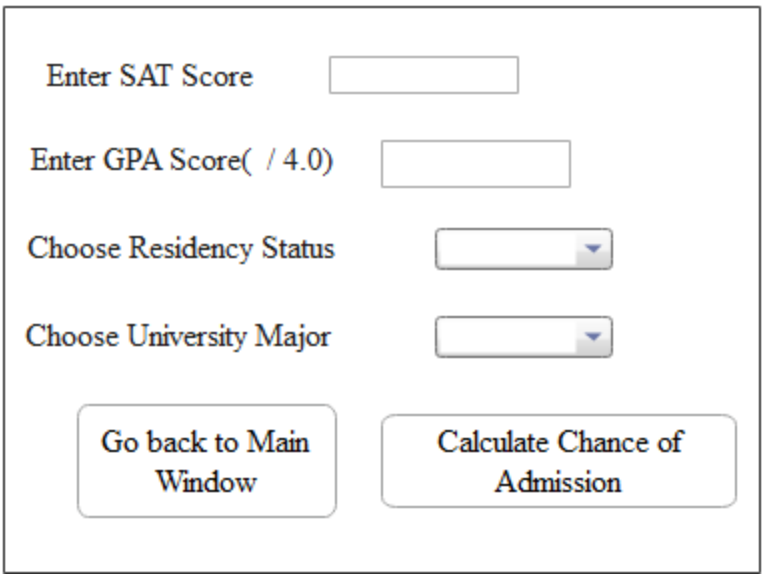


Fig 5

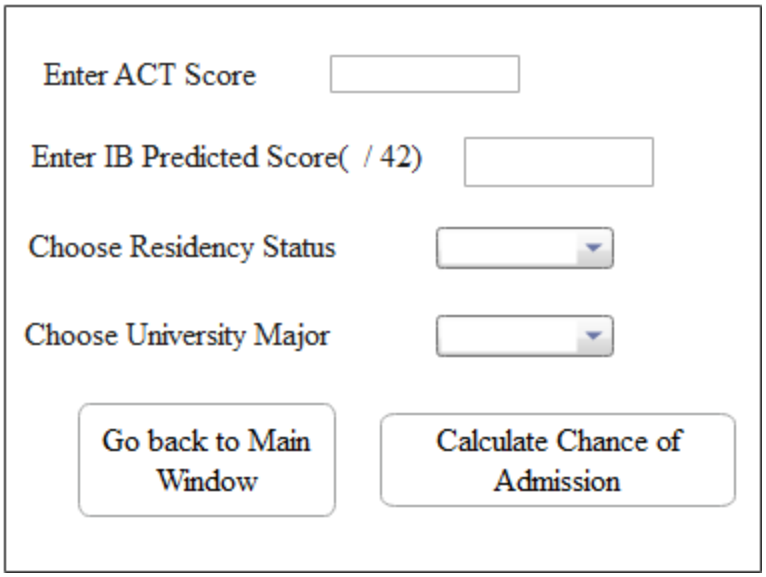
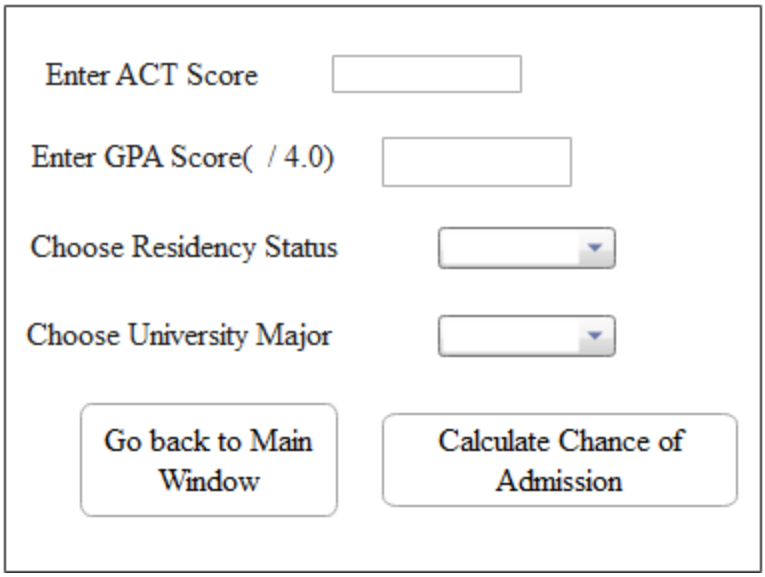


Fig 6



When a Private College is Chosen

Fig 7

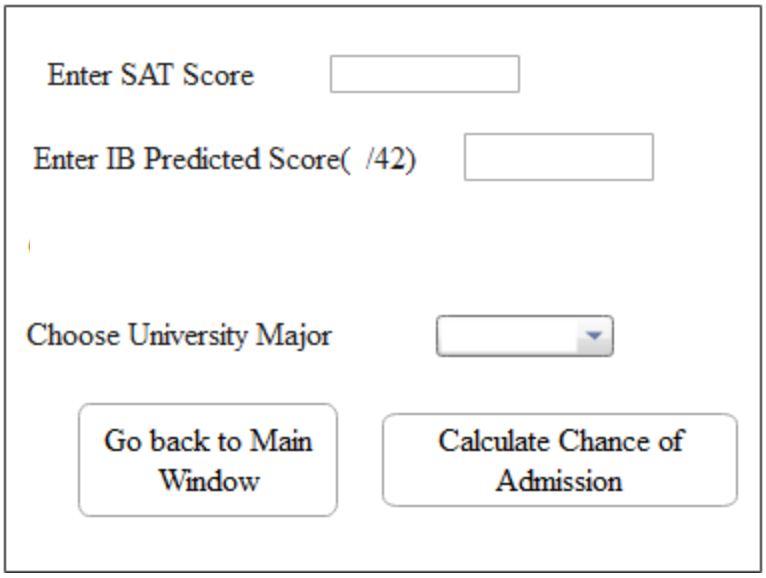


Fig 8

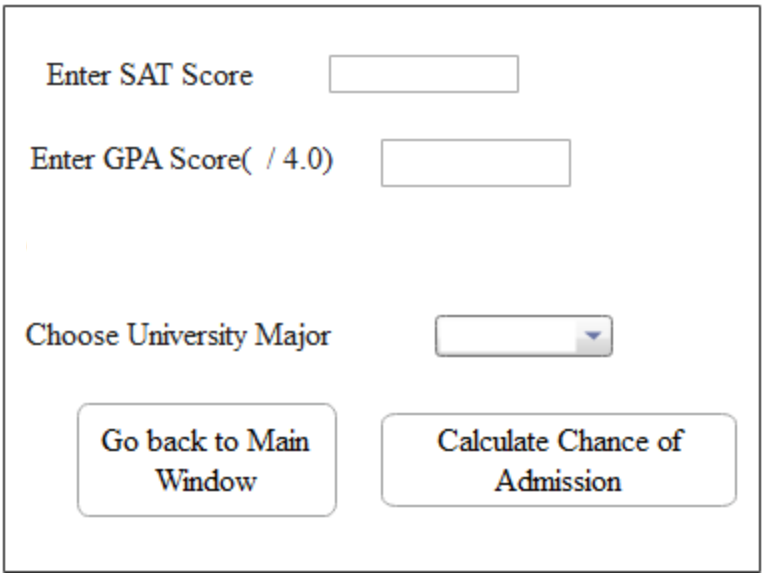


Fig 9

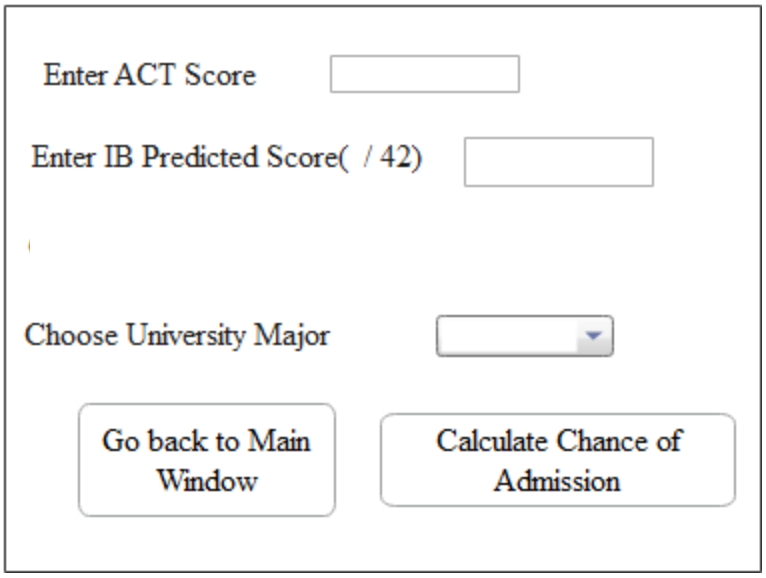
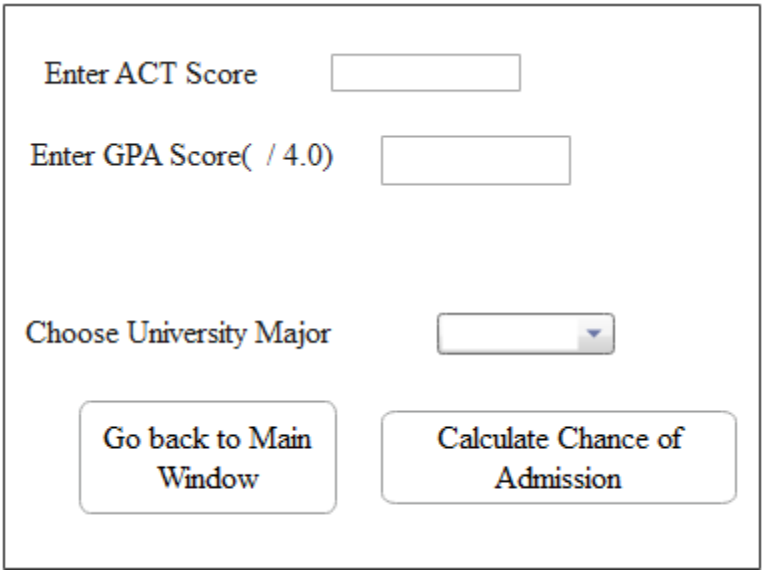
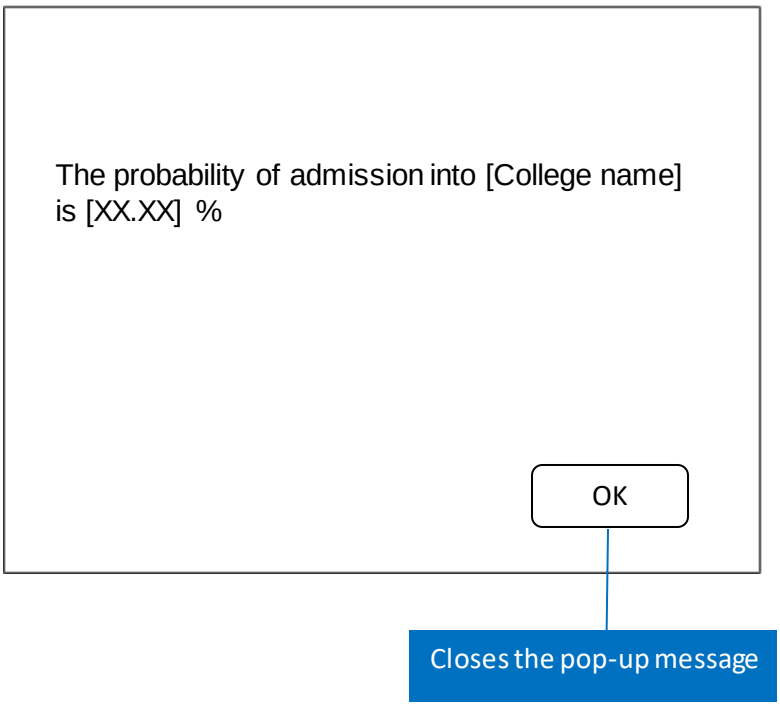


Fig 10



Pop Up Message



Test Plan

Test Number	Test description	Test data	Expected Outcome
1	Requirement 1: When the backend.exe file is launched, the cost function for the colleges should always decrease until convergence is met.	Training dataset for each college for which the cost function is to be minimized.	Convergence: Norm_Of_Projected_Gradient_<=_Pgtol i.e. convergence is met
2	Requirement 2: Automatically load the weights (Theta1 and Theta2) to be used for calculation based on user's selection of college, standardized test type and high school exam score type.	Choose any college, high school score type, standardized test type.	Correct path of Theta1.csv and Theta2.csv files based on the test data should be present in the 'log.txt' file.
3	Requirement 3: Automatically load list of majors (drop-down list) to choose choice of major based on user's selection of college.	Choose two colleges and any high school test, standardized score type.	Based on availability of majors of the college, regardless of the high school test, standardized score type, appropriate list should be displayed.

4	Requirement 4: Automatically load the menu to choose In-State/Out-of-State Status based on the college selected.	Choose a public college.	Radio button to choose In-state/Out-of-state residency should appear on the next screen.
5	Requirement 4: Automatically load the menu to choose In-State/Out-of-State Status based on the college selected.	Choose a private college.	Radio button to choose In-state/Out-of-state residency should not appear on the next screen.
6	Requirement 5: User should be able to input appropriate values for IB Predicted Score.	Enter a string or non-integer character.	A pop-up message should appear saying "Enter IB Predicted Score between 24 and 42"

7	Requirement 5: User should be able to input appropriate values for ACT Score.	Enter a string or non-integer character.	A pop-up message should appear saying "Enter ACT Score between 15 and 36"
8	Requirement 5: User should be able to input appropriate values for Predicted Percentage Score.	Enter a string or non-integer character.	A pop-up message should appear saying "Enter Percentage Score between 40 and 100"
9	Requirement 6: Validate user input for each parameter and display appropriate error in case validation returns false.	Enter a SAT Score not in the range of 720-1600	A pop-up message should appear saying "Enter SAT Score between 720 and 1600"

10	Requirement 6: Validate user input for each parameter and display appropriate error in case validation returns false.	Enter a IB Score not in the range of 24-42	A pop-up message should appear saying "Enter IB Predicted Score between 24 and 42"
11	Requirement 6: Validate user input for each parameter and display appropriate error in case validation returns false.	Enter an ACT Score not in the range of 24-42	A pop-up message should appear saying "Enter ACT Score between 15 and 36"
12	Requirement 6: Validate user input for each parameter and display appropriate error in case validation returns false.	Enter a Percentage Score not in the range of 40-100	A pop-up message should appear saying "Enter Percentage Score between 40 and 100"

13	Requirement 7: Based on inputs, calculate and display the probability of securing admission into user's choice of college.	Choose any public college, any high school test type, standardized score type, choice of major, and residency status.	A pop-up message should appear saying " The probability of admission into [College Name] is [XX.XX] %.
14	Requirement 7: Based on inputs, calculate and display the probability of securing admission into user's choice of college.	Choose any private college, any high school test type, standardized score type, choice of major.	A pop-up message should appear saying " The probability of admission into [University Name] is [XX.XX] %.