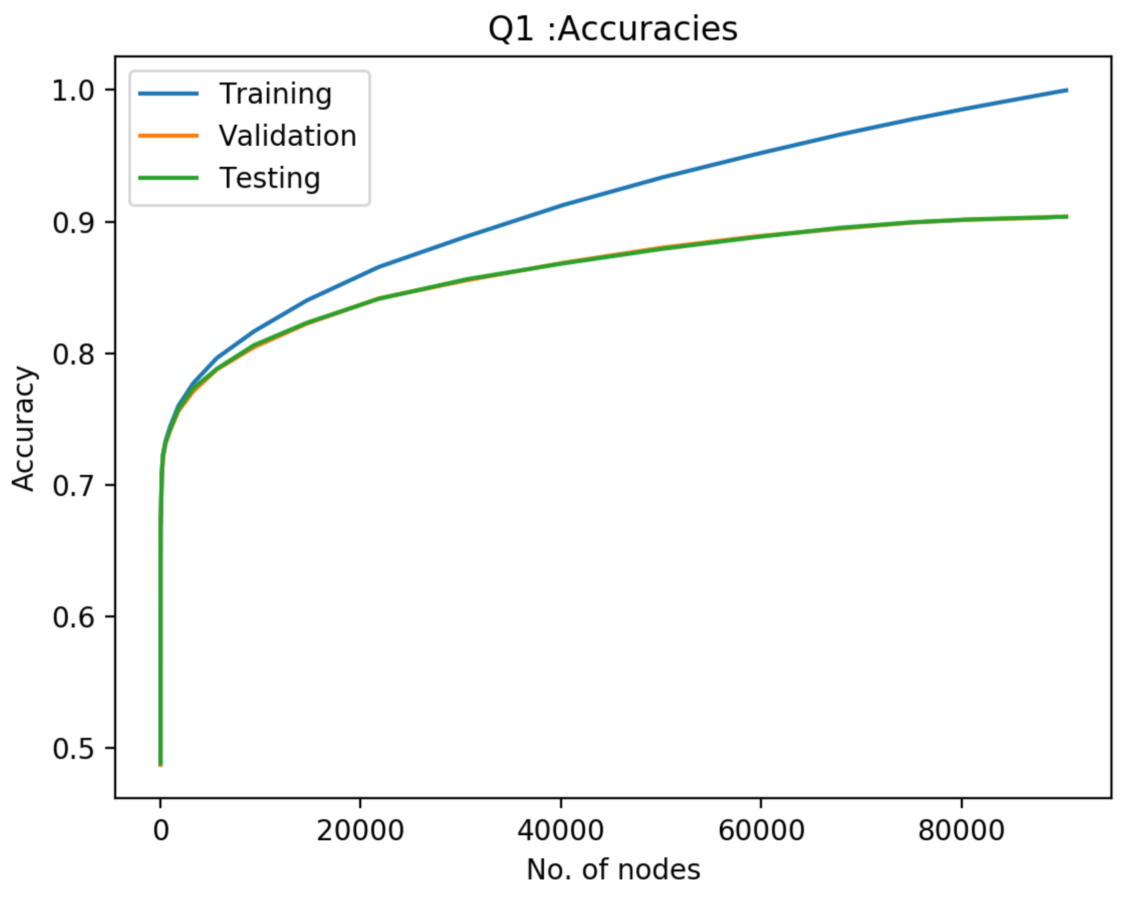
REPORT

Q1 : DECISION TREE

Part (a) :

- I observe from the graph that training accuracy increases more than the test and validation accuracies increase. This is because we are building the tree on the training data, so as the no. of nodes or the height of tree increases we are overfitting on the training data. So, the training accuracy approaches 100 % as we grow the tree larger and larger.
- With the increase in height of the tree we are fitting the noise in the training data, which is not likely to occur in the testing and validation data, hence the less increase in the latter two values.
- The graphs for testing and validation almost overlap probably due to the fact that both are of same type with respect to the data on which we trained. So in a way the tree is observing 2 new datasets of similar properties.

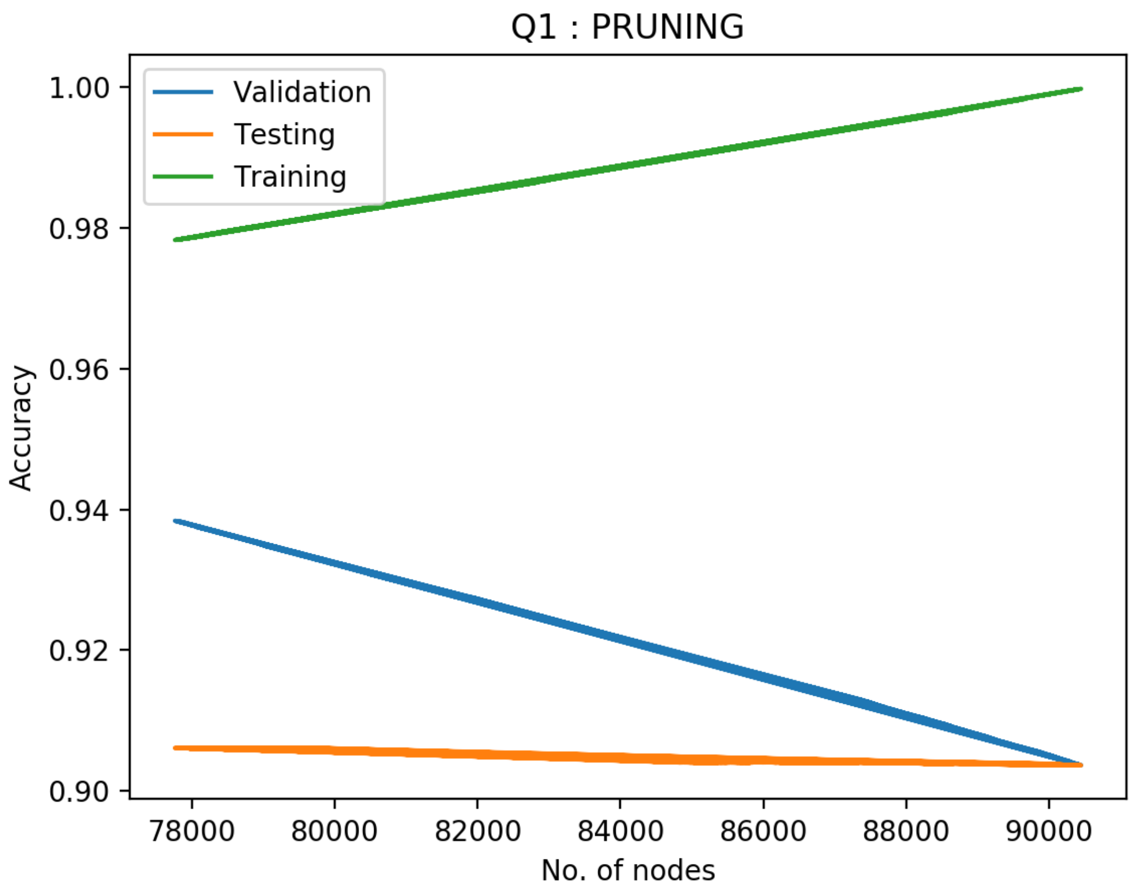
Training Data	=	99.975 %
Testing Data	=	90.363 %
Validation Data	=	90.362 %



<u>Part (b) :</u>

- The validation accuracy increases on pruning the tree from 90.36 % to 93.83 %. Test data accuracy remains almost same on pruning. The training data accuracy decreases on pruning the tree. Pruning was done using the validation data hence its expected to to increase. Test data is distributed in such a way that there is no significant effect of pruning on it. The tree was made using the training data itself due to which we had earlier overfit on the training data hence on pruning the accuracy will surely decrease.

Training Data	=	97.79 %
Testing Data	=	90.61%
Validation Data	=	93.83 %



Part (c) :

- I varied over these params, and got the data listed.

max_depth	: 1st column	[None, 29, 30, 31]
min_samples_leaf	: 2nd column	[1, 2, 3]
min_samples_split	: 3rd column	[2, 3, 4]

train accuracy	: 4th column
test accuracy	: 5th column
valid accuracy	: 6th column

None	1	2	100.0	92.968339888	92.9329959897
None	1	3	99.7527301517	92.9192877981	92.86673207
None	1	4	99.4607107717	92.8607695154	92.7849778833
None	2	2	98.6738648392	92.6215330069	92.5276673379
None	2	3	98.6741516952	92.5853893617	92.5182010637
None	2	4	98.6750122631	92.6473498963	92.5001290856
None	3	2	97.9730756984	92.4950302488	92.3443658457
None	3	3	97.9747968343	92.4674922334	92.371904098
None	3	4	97.9762311141	92.4967513747	92.3916972169
29	1	2	99.5568075225	92.8151596775	92.7505550679
29	1	3	99.3390838394	92.7850399732	92.735064801
29	1	4	99.0748894887	92.7007048011	92.6558923254
29	2	2	98.3497175903	92.5027753156	92.4243988916
29	2	3	98.35258615	92.531173894	92.4459131512
29	2	4	98.3474227425	92.4640499815	92.4218171804
29	3	2	97.6931042693	92.3229176527	92.2238859916
29	3	3	97.6965465409	92.35389792	92.2565876663
29	3	4	97.696259685	92.3392683494	92.2006505912
30	1	2	99.6833110064	92.924451176	92.8228429803
30	1	3	99.457555356	92.8323709371	92.8073527134
30	1	4	99.1859027501	92.7531991429	92.7522762087
30	2	2	98.4406509336	92.5242893901	92.4768936851
30	2	3	98.4504040366	92.5225682642	92.4794753963
30	2	4	98.4438063493	92.5681781021	92.5070136486
30	3	2	97.7820296207	92.4167190176	92.3176881637
30	3	3	97.7826033327	92.4029500099	92.3125247414
30	3	4	97.7840376126	92.4081133878	92.2660539406
31	1	2	99.7891608602	92.9812483327	92.8856646185
31	1	3	99.5585286583	92.880562464	92.855544655
31	1	4	99.2842943487	92.8237653073	92.7608819125
31	2	2	98.5321579888	92.5896921766	92.5156193525
31	2	3	98.5255603014	92.5819471098	92.4777542555
31	2	4	98.5249865895	92.5836682358	92.5293884787
31	3	2	97.8462853586	92.3831570613	92.3168275933
31	3	3	97.8480064944	92.44339647	92.3245727268
31	3	4	97.8503013422	92.4597471666	92.3684618165

Validation Accuracies :
Decrease with increase in min_samples_split.
Decrease with increase in min_samples_leaf.
Increase with increase in max depth.

With all three of the above variations since we are heading towards pure leafs, maybe the validation set is such that it was better classified using more pure leaves, and without restricting its height.

The best validation accuracy I got is less than accuracy in b part obtained after pruning. Its possibly because the pruning was done using the validation data set itself. The training and test data accuracy are better here than b part since pruning didn’t help the test data much, and surely didn't help the training data.

<u>Maximum validation accuracies are obtained for</u>

max_depth	:	None
min_samples_leaf	:	1
min_samples_split	:	2

<u>And the accuracies are</u>

Training	:	100 %
Test	:	92.968339888 %
Validation	:	92.9329959897 %

Part (d) :

- I varied over these params, and got the data listed.

n_estimators	: 1st column	[8, 10, 12, 14]
max_features	: 2nd column	[None, auto, sqrt]
bootstrap	: 3rd column	[True, False]

train accuracy	: 4th column
test accuracy	: 5th column
valid accuracy	: 6th column

Validation Accuracies :
Increase with increase in n_estimators.
Decrease with any other than None type of max_features.
Increases for True value if Bootstrap parameter.

The accuracy increases with increase in the number of the decision trees in the forest because each different tree will be built using slightly different randomised input data and hence randomised split points. So by averaging the predictions of all the trees, we get a stronger prediction and minimise overfitting. Hence this is also the reason for the higher accuracies in this part than all above parts, where the lone tree ends up learning lots of rules which are specific to the quirks of the training data.

Maximum validation accuracies are obtained for

n_estimators	: 14
max_features	: None
bootstrap	: True

And the accuracies are

Training	: 99.88 %
Test	: 95.84 %
Validation	: 95.85 %

8	None	True	99.6640916562	95.2953021867	95.2487908986
8	auto	True	99.5972542146	92.9709215769	92.9398805528
8	sqrt	True	99.6110233013	93.1568031806	92.9854907833
8	None	False	100.0	93.1886440109	93.0758506738
8	auto	False	100.0	94.1714069344	94.1859864718
8	sqrt	False	100.0	94.2531604175	94.2255727096
10	None	True	99.770228366	95.5698217774	95.5095437256
10	auto	True	99.7610489749	93.3667805478	93.2505464622
10	sqrt	True	99.7607621189	93.5836424189	93.4553622141
10	None	False	100.0	93.1765961292	93.127484897
10	auto	False	100.0	94.4355997694	94.2608560954
10	sqrt	False	100.0	94.5078870597	94.3391680006
12	None	True	99.833049824	95.6205949932	95.6988692105
12	auto	True	99.8353446718	93.8977479067	93.8202440578
12	sqrt	True	99.8370658076	93.7024001102	93.5130204299
12	None	False	100.0	93.228229908	93.1360906009
12	auto	False	100.0	94.626644751	94.5956179756
12	sqrt	False	100.0	94.670533463	94.5147243593
14	None	True	99.8832496192	95.8426202422	95.8589353023
14	auto	True	99.8935764342	93.8134127346	93.8219651985
14	sqrt	True	99.8915684424	93.9089352254	93.8615514363
14	None	False	100.0	93.2454411676	93.1524414382
14	auto	False	100.0	94.715282738	94.6997469923
14	sqrt	False	100.0	94.8305981773	94.6541367618

Q2 : **NEURAL NETWORKS**

Part (a) :

- I chose **eta = 0.1** as my learning rate
- My stopping criteria was to check if the difference between 2 consecutive cost values (J) is less than threshold **d = 0.001**, and that too for 3 consecutive times. If this holds then I break out from the while loop of Stochastic Gradient Descent.

TRAINING DATA	: **79.12%** Accuracy
TESTING DATA	: **71.40%** Accuracy

- Time taken to learn parameters is : **183.5 seconds**
- Total no. of iterations taken : 34 * m

Part (b) : eta varied

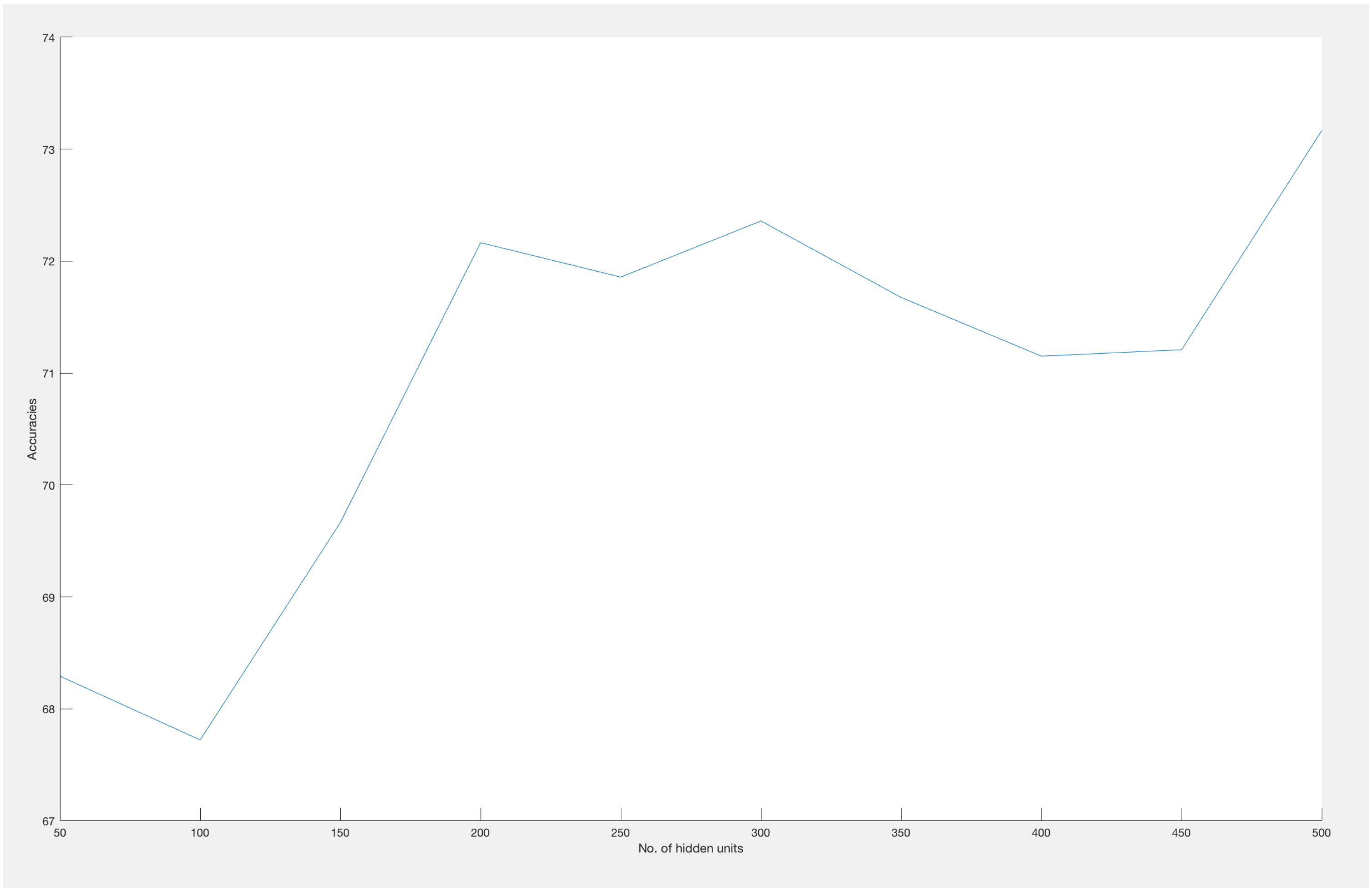
- I varied eta as equal to 0.3 ÷ √(iter).
- Convergence Rate increases as no. of iterations decrease to 20. This thing would also depend on the constant factor used in that proportion above.

TRAINING DATA	: **80.43%** Accuracy
TESTING DATA	: **71.98%** Accuracy

- Accuracies increase slightly as compared to part (a). So initially as the gradient descent starts it takes bigger steps which is more intuitive since we are far way from the local optimum, so it saves time and reaches closer to the local optimum in less number of iterations. And as we get closer to the optimum with increase in number of iterations, the learning rate is decreased, hence taking smaller steps, which is the right thing to so as we don't want to miss the optimum and shoot away from it due to larger steps. Hence we get slightly better accuracy than before.

Part (c) : No. of hidden units varied

- The convergence criteria was a bit relaxed since running for > 250 hidden units was taking lot of time. So this is the first thing I observed that it takes a lot of time learn parameters by back propagation if hidden layer has more number of units of perceptron.
- Now as we increase the number of units we see there is an increase in accuracy on average (as graph contains zig zags in between), it peaks in between and then decrease for more larger number of units. This is possibly because we are doing overfitting on the training data, which is not reflected in positive way on the test data.
- It also depends on the type of problem we are solving and the type of data available for it.



Part (d) : Softplus function

- For **d = 0.001** takes **18** iterations to converge.
- Time taken to converge is **229.35 seconds**.
- Accuracy on training data : **76.226 %**
- Accuracy on test data : **68.024 %**
- This softplus function is not a bounded one, it produces large outputs if the weights are high enough, hence had to initialise the params very very close to zero to avoid getting infinity values in between the algorithm.
- It doesn't work better than the sigmoid case possibly because the fact that the sigmoid function behaves as a linear function close to inputs around zero, but the softplus function doesn't. And we want this property of linear function since we originally had the linear function at the end of a perceptron to classify the input into 0 or 1 depending on input is +ve or -ve. Moreover the sigmoid function lies bw 0 and 1 and gives us a sense of probability about which of the 2 cases is how much probable but the soft pus function has its range as all real numbers hence won't behave as good as sigmoid was doing.