CSE41 NEURAL NETWORKS

REPORT ON

PREDICTING THE MEDIAN VALUE OF HOUSE PRICING
&
PREDICTING THE PRESENCE OF HEART DISEASE

By

Afroz Hudli
14982917

38/40 95%

Submitted on 02.05.05

Afroz,
An excellent report. You have done more than you were required to do, especially in regards to the advanced feature preprocessing steps you performed. Indeed, the test accuracies in Part II are very high. I would have liked a bit more explanation on why you have measured performance using the correlation coefficient — this was not clear to me. Nevertheless — excellent work!

Abstract

The Multilayer Perceptrons better known as (MLPs) are the subject of study of this report. The report uses two data sets namely the Boston data set and Cleveland data set which are both real world data sets, for investigation. These data sets have been used to understand how the different parameters like network size, learning rate, momentum, learning algorithms, weight regularization, early stopping influence the generalization capabilities of the network.

1. INTRODUCTION

The scope of the report is to better understand the Multilayer Perceptrons (MLPs), an important class of Neural Networks. The MLP network consists of a set of sensory units (Source nodes) that constitute the ***input layer***, ***one or more hidden layers*** of computational nodes, and an ***output layer*** of computational nodes. MLPs have been applied successfully to solve regression and classification problems by training them in a supervised manner with a highly popular algorithm known as the **error back-propagation algorithm**. The error back-propagation learning consists of two passes through the different layers of the network: ***a forward pass*** and ***a backward pass***.

In the forward pass, an input vector is applied to the sensory nodes of the network, and its effect propagates through the network layer by layer. Finally, a set of outputs is produced as the actual response of the network. During the forward pass the synaptic weights of the networks are all fixed.

During the backward pass, the synaptic weights are all adjusted in accordance with an error-correction rule. The actual response of the network is subtracted from a target response to produce an error signal. This error signal is then propagated backward through the network, against the direction of synaptic connections, hence the name back-propagation. The synaptic weights are adjusted to make the actual response of the network move closer to the desired response in a statistical sense.

$\Delta w_{ji}(n) = \eta \delta_j(n) y_i(n)$

where η is the learning-rate parameter of the back-propagation algorithm. The smaller the learning rate parameter, the smaller the changes to the synaptic weights in the network. If, the learning rate parameter is too large, the resulting large changes in the synaptic weights assume such a form that the network becomes unstable. To avoid the danger of instability, the delta rule is modified by including a momentum term,

$\Delta w_{ji}(n) = \alpha \Delta w_{ji}(n-1) + \eta \delta_j(n) y_i(n)$

where α is usually a positive number called the momentum constant. The momentum term also has the benefit of preventing the learning process from terminating in a shallow local minimum on the error surface.

As the learning process progresses, the hidden neurons begin to gradually "discover" the salient features that characterize the training data. They do so by performing a nonlinear transformation on the input data into a new space called the hidden space, or feature space.

Stopping Criteria: The back propagation algorithm is considered to have converged when the absolute rate of change in the mean squared error per epoch is sufficiently small.

The report consists of two parts: Part1, a regression problem, and Part 2, a classification problem. In Part1, I have investigated how the of network size(number of hidden units), learning algorithms, training parameters, early stopping, and weight regularization involved in MLP training affect the performance of the network on the Boston Housing dataset. In Part2, I have investigated the performance of MLPs as binary classifiers. I had the option of conducting the investigation in Matlab or Netlab. I choose to work with both Matlab and Netlab, so as to give me an opportunity to familiarize myself with the two tools. Thus in Part1 Matlab was used to investigate the performance of the network for the batch back-propagation Algorithm, and Netlab for the Scaled Conjugate Gradient Algorithm. Part2 was entirely done in Netlab.

2. PART1: PREDICTING THE MEDIAN VALUE OF HOUSE PRICING

2.1 METHODOLOGY:

Harrison and Rubinfeld (1978) undertook a hedonic pricing study based on data from the 1970 census of the Boston Standard Metropolitan Statistical Area. Hedonic pricing method is that the price of a marketed good is related to its characteristics, or the service it provides[1]. This method consists of two steps; first data collection and second Regression analysis. Here this regression analysis is aided by a MLP. We were provided with 506 samples of data containing 14 features divided into three sets: Training, Validation, and Test set. Also the data set given was already preprocessed.

1. No of Training Inputs Required:

We have 13 inputs i.e 13 dimensions. For high dimensions the curse of dimensionality holds. There is an exponential growth in complexity as a result of an increase in dimensionality. The data set required increases exponentially. For a good generalization, the size of training set, N, should satisfy the condition:

$N = O(W/\epsilon)$

W = free parameters(Synaptic weights and biases) and ϵ fraction of error permitted.

With 13 inputs and a single hidden layer with 4 units we have W=76 and say if $\epsilon = 10\%$ then, N = 760. But the training set given is of size 253. Thus it is important to reduce the inputs.

To achieve the same, I used the original data (from web. See Appendix) to understand the data and how each feature relates with the 14^{th} feature. The correlation coefficients were found to be:

CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
1	-0.2	0.41	-0.1	0.42	-0.22	0.35	-0.38	0.63	0.58	0.29	-0.39	0.46	-0.39
-0.2	1	-0.53	-0	-0.5	0.31	-0.57	0.66	-0.31	-0.31	-0.39	0.18	-0.41	0.36
0.41	-0.53	1	0.06	0.76	-0.39	0.64	-0.71	0.6	0.72	0.38	-0.36	0.6	-0.48
-0.06	-0.04	0.06	1	0.09	0.09	0.09	-0.1	-0.01	-0.04	-0.12	0.05	-0.05	0.18
0.42	-0.52	0.76	0.09	1	-0.3	0.73	-0.77	0.61	0.67	0.19	-0.38	0.59	-0.43
-0.22	0.312	-0.39	0.09	-0.3	1	-0.24	0.21	-0.21	-0.29	-0.36	0.13	-0.61	0.7
0.35	-0.57	0.64	0.09	0.73	-0.24	1	-0.75	0.46	0.51	0.26	-0.27	0.6	-0.38
-0.38	0.664	-0.71	-0.1	-0.8	0.21	-0.75	1	-0.49	-0.53	-0.23	0.29	-0.5	0.25
0.63	-0.31	0.6	-0	0.61	-0.21	0.46	-0.49	1	0.91	0.46	-0.44	0.49	-0.38
0.58	-0.31	0.72	-0	0.67	-0.29	0.51	-0.53	0.91	1	0.46	-0.44	0.54	-0.47
0.29	-0.39	0.38	-0.1	0.19	-0.36	0.26	-0.23	0.46	0.46	1	-0.18	0.37	-0.51
-0.39	0.176	-0.36	0.05	-0.4	0.13	-0.27	0.29	-0.44	-0.44	-0.18	1	-0.37	0.33
0.46	-0.41	0.6	-0.1	0.59	-0.61	0.6	-0.5	0.49	0.54	0.37	-0.37	1	-0.74
-0.39	0.36	-0.48	0.18	-0.4	0.7	-0.38	0.25	-0.38	-0.47	-0.51	0.33	-0.74	1

The scatter plots of the 13 features were also plotted against the MEDV (See Appendix A for plots). We see that six parameters namely ZN, INDUS, RM, PTRATIO, B, and LSTAT show a clear relation with MEDV (see Appendix B for list of variables used) .

There after I performed the ***Principal Component Analysis*** which uncorrelated (removes redundancies) the input variables. This was done using *prepca* function in Metlab. Thus reducing the dimensions of the input vector. This technique has three effects: it orthogonalizes the components of the input vectors (so that they are uncorrelated with each other); it orders the resulting orthogonal components (principal components) so that those with the largest variation come first; and it eliminates those components that contribute the least to the variation in the data set. It is important to normalize the input vector so that they have zero mean and unity variance. This was done by using prestd. I

have retained those principal components which account for 95% of the variation in data set[+]. The no of components which account for 95% of variations were found to be 6. Now, with 6 inputs and N= 253 ϵ = 13%, which is much better than ϵ =24%.

2. Optimal Number of Hidden Neurons:

The optimal number of Hidden Neurons is one for which a ***good generalization*** is achieved for the test data. To estimate the same, I set the learning rate, momentum, number of epochs, and recorded the correlation coefficients between n/w training_t & training_t, n/w vali_t & vali_t and n/w test_t & test_t using the *routine postreg*.

3. Learning Algorithms:

The scope of this report was limited to the investigation of the Back Propagation Algorithm and Scaled Conjugate Gradients Algorithm. This was achieved by setting the function to 'traingd'(without momentum)/ 'traingdm' (with momentum) in Metlab for Back Propagation and 'scg' in Netlab for Scaled Conjugate Gradients. Comparison between the MSE of the networks trained by the different algorithms for same number of epochs was made.

4. Training Parameters:

The optimal learning rate and momentum where investigated by setting all other parameters constant and varying first the learning rate and then momentum.

5. Early Stopping:

In Metlab the early stopping is incorporated as a network parameter. Thus validation stop was encountered when the validation error started to rise. For Netlab I plotted the MSE Vs Epochs for the training, validation and test data and made note of the test error for the epoch corresponding to the point on validation curve where the error starts to increase, as beyond this point the network is learning noise contained in the training data.

[+] See Appendix C for details of how I can to the conclusion of retaining 95% components.

6. Weight Regulation:

Weight regularization is done to achieve a good generalization. A penalty term based on the magnitude of the weights is introduced, causes gradient descent to seek vectors with small magnitudes. Smaller network weights imply less sensitivity to input change. This was done by setting 'prior' in Netlab.

Also I have not initialized the weights using a 'seed'. This was done to eliminate any biases introduced if any. Both Netlab as well as Metlab initialize the weights randomly. But between successive runs for the same parameters I did reinitialize the weights using '*mlpinit*' for netlab and '*init*' for Metlab.

2.2 Results & Conclusion:

2.2.1 Network Size:

Boston Data Back Propagation with Momentum:

LEARNING RATE	MOMENTUM	HIDDEN NEURONS	CORRELATION COEFFICIENT BETWEEN N/W TRAINING_T & TRAINING_T	CORRELATION COEFFICIENT BETWEEN N/W VALI_T & VALI_T	CORRELATION COEFFICIENT BETWEEN N/W TEST_T & TEST_T
0.01	0.05	30	0.969	0.855	0.86
		13	0.934	0.847	0.832
		8	0.917	0.829	0.821
		4	0.895	0.813	0.806

Boston Back Propagation without Momentum:

LEARNING RATE	EPOCHS	HIDDEN NEURONS	CORRELATION COEFFICIENT BETWEEN N/W TRAINING_X & TRAINING_T	CORRELATION COEFFICIENT BETWEEN N/W VALI_X & VALI_T	CORRELATION COEFFICIENT BETWEEN N/W TEST_X & TEST_T
0.5	1000	13	-0.417	-0.33	-0.273
0.5	1000	10	0.0557	-0.073	-0.0198
0.001	10000	10	0.842	0.75	0.715
0.001	10000	13	0.868	0.797	0.776
0.01	1000	5	0.92	0.837	0.829
0.01	1000	2	0.796	0.737	0.709

Conclusion: We see that the correlation coefficients are higher for higher number of hidden neurons. But this also means that the higher the number of hidden neurons the greater is the chance of over fitting where as if the number is too small (say 2 in this case) underfitting occurs. Thus from table above, 5 hidden neurons seam to give good generalization.

2.2.2 Learning Algorithms:

Back Propagation:
Hidden Neurons:13; Learning Rate:0.5;
TRAINGD, Epoch 0/1000, MSE 3.78203/1e-005, Gradient 9.88581/1e-010
TRAINGD, Epoch 6/1000, MSE 484847/1e-005, Gradient 10388.1/1e-010
TRAINGD, Validation stop.

Hidden Neuron:10,LR:0.001,Epochs:10000,
TRAINGD, Epoch **1000**/10000, **MSE 0.205413**/1e-005, Gradient 0.195116/1e-010
TRAINGD, Epoch 10000/10000, **MSE 0.119905**/1e-005, Gradient 0.0531238/1e-010

Scaled Conjugate Gradient: Epochs:1000

HIDDEN NEURONS	MSE VALIDATION ERROR	MSE TEST ERROR
13	0.014095	0.013818
10	0.013241	0.013822

Conclusion: We see that the MSE test error is much smaller in SCG for the same number of hidden neurons. Thus one could conclude that the 'scg' is indeed a faster algorithm as compared to back propagation.

2.2.3Training Parameters:

Learning Rate: Boston Graddesc

LEARNING RATE	EPOCHS	HIDDEN NEURONS	CORRELATION COEFFICIENT BETWEEN TRAINING_X & TRAINING_T	CORRELATION COEFFICIENT BETWEEN VALI_X & VALI_T	CORRELATION COEFFICIENT BETWEEN TEST_X & TEST_T
0.001	1000	13	0.868	0.797	0.776
0.01	1000	13	0.937	0.861	0.832
0.5	1000	13	-0.417	-0.33	-0.273
0.001	1000	10	0.842	0.75	0.715
0.01	1000	10	0.958	0.872	0.855
0.5	1000	10	0.0557	-0.073	-0.0198

Conclusion: If the learning rate is too high(0.5) or too small (0.001) the network response is poor. With learning rate = {0.5,0.9}, network becomes unstable.
It appears that learning rate of 0.01 is an optimal value in this case.

Avoid claiming that it is 'optimal'.
Better to say something like 'appropriate or suitable'.

Momentum:

Boston Graddsec:

LEARNING RATE	MOMENTUM	HIDDEN LAYER	CORRELATION COEFFICIENT BETWEEN N/W TRAINING_T & TRAINING_T	CORRELATION COEFFICIENT BETWEEN N/W VALI_T & VALI_T	CORRELATION COEFFICIENT BETWEEN N/W TEST_T & TEST_T
0.01	0.001	5	0.928	0.846	0.831
	0.01		0.924	0.822	0.794
	0.1		0.915	0.83	0.83
	0.99		0.441	0.308	0.356

Conclusion: If the momentum is too high(0.99) the network response is poor. With momentum = 0.9, network becomes unstable.
It appears that momentum of 0.1 is an optimal value in this case.

It was also seen that for learning rate ->0, the use of momentum ->1 produces increasing speed convergence and for learning rate ->1, the momentum -> 0 is required to ensure learning stability.

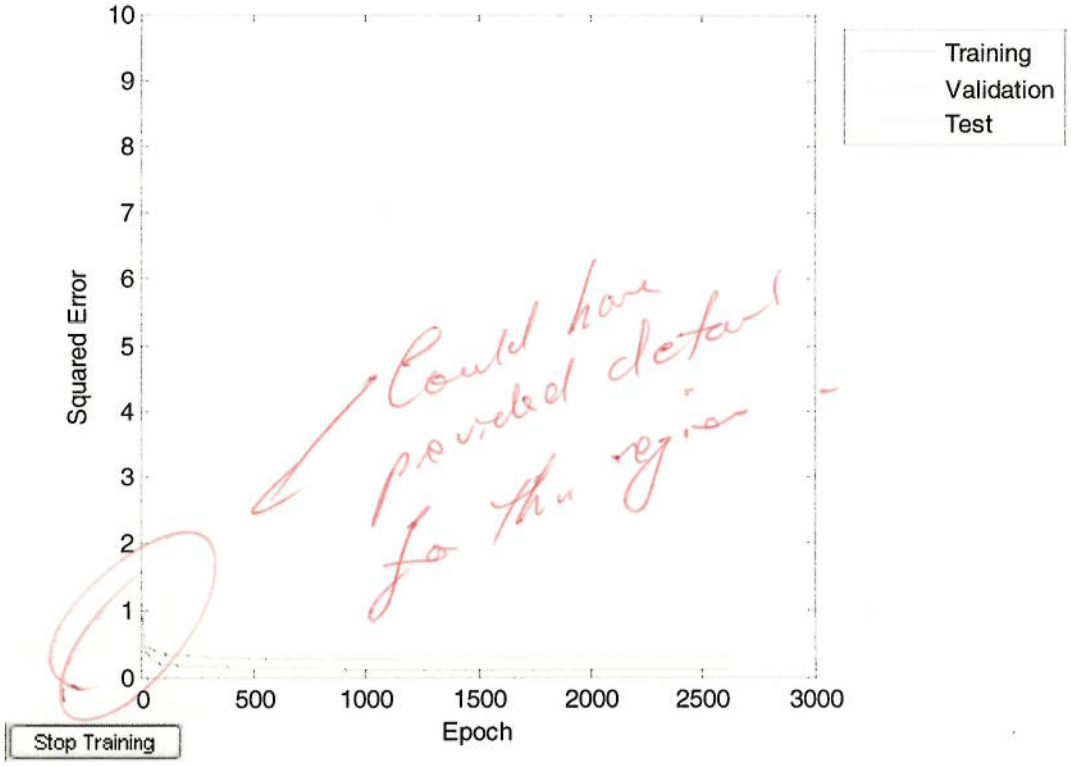
Epochs:

LEARNING RATE	EPOCHS	HIDDEN LAYER	CORRELATION COEFFICIENT BETWEEN TRAINING_X & TRAINING_T	CORRELATION COEFFICIENT BETWEEN VALI_X & VALI_T	CORRELATION COEFFICIENT BETWEEN TEST_X & TEST_T
0.01	1000	13	0.937	0.861	0.832
0.01	10000	13	0.952	0.866	0.854
0.001	1000	10	0.842	0.75	0.715
0.001	10000	10	0.938	0.848	0.838

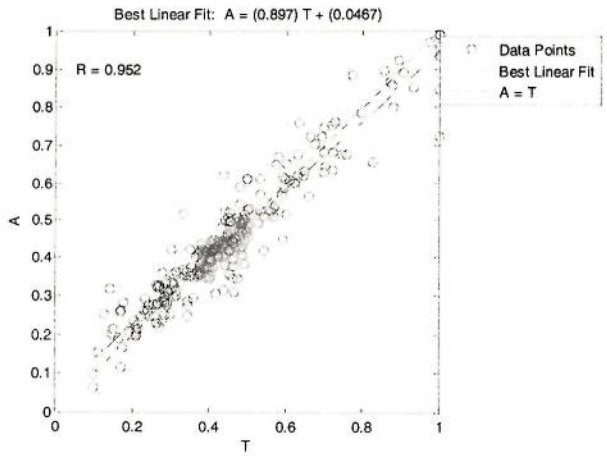
Conclusion: We see that with increasing number of epochs the correlation coefficient increases. This in other words, is leading towards overfitting. For good generalization, the number of epochs should be just sufficient so that the network learns and not memorizes.

2.2.4 Early stopping:

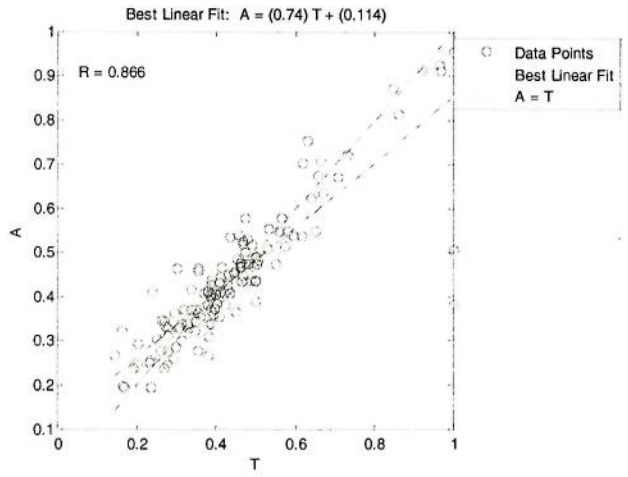
Hidden Neurons = 13; Learning Rate = 0.01; Epochs = 10000;



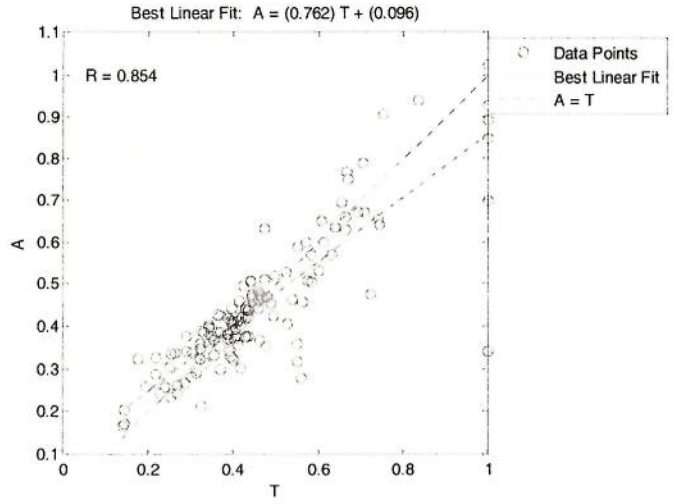
TRAINGD, Epoch 2500/10000, MSE 0.0941079/1e-005, Gradient 0.0238702/1e-010
TRAINGD, Epoch 2550/10000, MSE 0.0938266/1e-005, Gradient 0.0235638/1e-010
TRAINGD, Epoch 2600/10000, MSE 0.0935523/1e-005, Gradient 0.0232839/1e-010
TRAINGD, Epoch 2628/10000, MSE 0.0934014/1e-005, Gradient 0.0231371/1e-010
TRAINGD, Validation stop.



correlation coefficient between n/w training_t & training_t



correlation coefficient between N/w vali_t & vali_t



correlation coefficient between n/w test_t & test_t

Conclusion: The network will overfit if training session is not stopped at the right point. Here the training was stopped at 2628th epoch. This was the point were the validation curve started to increase. Beyond this point the network will actually be learning noise.

From the graph on the previous page, it appears that the errors have stabilized well before this point!

2.2.5 Weight Regularization:

Boston SCG:

HIDDEN LAYER	PRIOR	MSE VALIDATION ERROR	MSE TEST ERROR
5	0	0.0090111	0.0076361
	0.01	0.0073913	0.010101
30	0	0.0221089	0.02991
	0.01	0.0064639	0.0079264
	0.1	0.0078123	0.0072527
	10	0.011514	0.013933

Conclusion: With no 'prior' set the network overfits. When a small value (say 0.01) is set the network actually learns just enough leading to good generalization. This can be verified from table above. When the value of prior is very high the network underfits.

3 PART2: PREDICTING THE PRESENSE OF HEART DISEASES

3.1 METHODOLOGY:

We were provided with 303 samples of data containing 14 features. The objective is to predict the presence or absence of heart disease, i.e classify a given input in either the class 'heart disease present' or class 'heart disease absent'. The data given had some samples with missing values.

1. Preprocessing:

First the missing values for samples, was dealt with. I had the option of totally discarding the samples. But as the input is of 13 dimensions and available data set is only of size 303 I decided to retain the samples. I simply replaced the missing values by the mean values for that feature. This was done in excel, though could have been done in Matlab using the mean function. Then I scaled the network inputs and targets to normalize the mean and standard deviation of the training set. This procedure is implemented in the function *prestd*. It normalizes the inputs and targets so that they will have zero mean and unity standard deviation.

2. No of Training Inputs Required:

As discussed in **Part 1** for a good generalization, the size of training set, N, should satisfy the condition:

$N = O(W/\epsilon)$

W = free parameters(Synaptic weights and biases) and ϵ fraction of error permitted.

With 13 inputs and a single hidden layer with 4 units we have W=76 and say if ϵ = 10% then, N = 760. But the training set given is of size 303. Thus it is important to reduce the inputs.

To achieve the same I performed the ***Principal Component Analysis*** which uncorrelated (removes redundancies) the input variables. Thus reducing the dimensions of the input vector. As discussed in **Part1** 2.1, this was done by using *prepca*. I have retained those principal components which account for 94% of the variation in data set^{+}. The no of components which account for 94% of variations were found to be 7.

Now, with 7 inputs and N= 303 ϵ = 12%, which is much better than ϵ =25%.

3. 5 Fold Cross Validation:

The entire data set of 303 samples can not be used for training as we will need samples for test data set too. Thus, to provide the network with sufficient number of training samples the data set was divided into five almost equal size sub sample sets. Of which 4 were used for training and the last set was used for testing. Care was take to see that the samples in each of the five sets was randomly chosen (see code for details). I then divided the training data into two subgroups of training set and validation set. Thus the size of training set came out to be 242 i.e 15% error.

4. Optimal Number of Hidden Neurons:

The optimal number of Hidden Neurons is one for which a ***good generalization*** is achieved for the test data. To estimate the same, I set the learning rate, momentum, number of epochs, and recorded the accuracy for test, validation and training data.

5. Learning Algorithms:

The scope of this report was limited to the investigation of the Back Propagation Algorithm and Scaled Conjugate Gradients Algorithm. This was achieved by setting the function to 'graddesc' in Netlab for Back Propagation and 'scg' in Netlab for Scaled Conjugate Gradients. Comparison between the Accuracy of the networks trained by the different algorithms for same number of epochs was made.

6. Training Parameters:

The optimal learning rate and momentum where investigated by setting all other parameters constant and varying first the learning rate and then momentum.

7. Weight Regulation:

Weight regularization is done to achieve a good generalization. A penalty term based on the magnitude of the weights is introduced, causes gradient descent to seek vectors with small magnitudes. Smaller network weights imply less sensitivity to input change. This was done by setting 'prior' in Netlab.

Also I have not initialized the weights using a 'seed'. This was done to eliminate any biases introduced if any. Both Netlab as well as Metlab initialize the weights randomly. But between successive runs for the same parameters I did reinitialize the weights using '*mlpinit*' for netlab and '*init*' for Metlab.

3.2 Results & Conclusion:

3.2.1 Network Size:

LEARNING RATE	MOMENTUM	HIDDEN LAYER	TEST ACCURACY	VALIDATION ACCURACY	TRAINING ACCURACY
0.001	0.05	30	87.087	82.874	93.728
		20	86.754	83.858	91.858
		10	84.781	79.913	91.416
		6	83.131	81.541	88.117
		2	80.169	80.53	83.278
		40	84.432	84.536	93.616

Cleveland Data Graddesc

HIDDEN NEURONS	TEST ACCURACY	VALIDATION ACCURACY	TRAINING ACCURACY
13	85.459	82.896	94.825
10	84.104	80.0891	93.286
8	81.47	82.208	92.627
6	81.787	80.557	89.882
4	81.82	78.918	89.769
2	78.852	79.557	84.153
30	84.749	83.208	94.605

CSCG

Conclusion: With increase in hidden layers the response of the network improves till a point and there after any increase in the hidden layers actually decreases the generalization performance. This is verified by the fact that for 40 hidden neurons in table1 above, the test accuracy is less than for 20 hidden neurons.

3.2.2 Learning Algorithm:

LEARNING RATE	MOMENTUM	HIDDEN LAYER	TEST ACCURACY	VALIDATION ACCURACY	TRAINING ACCURACY
0.001	0.05	30	87.087	82.874	93.728
		20	86.754	83.858	91.858
		10	84.781	79.913	91.416
		6	83.131	81.541	88.117
		2	80.169	80.53	83.278
		40	84.432	84.536	93.616

Cleveland Data Graddsec: Epochs =1000

HIDDEN LAYER	PRIOR	TEST ACCURACY	VALIDATION ACCURACY	TRAINING ACCURACY
13		85.459	82.896	94.825
10		84.104	80.0891	93.286
8		81.47	82.208	92.627
6		81.787	80.557	89.882
4		81.82	78.918	89.769
2		78.852	79.557	84.153
30		84.749	83.208	94.605
30	10	82.18	80.852	82.945

CSCG: Epochs = 100

Conclusion: The back propagation algorithm is generally very slow because it requires small learning rates for stable learning. The momentum variation is usually faster than simple gradient descent, since it allows higher learning rates while maintaining stability, but it is still too slow for many practical applications. As against this, scaled conjugate gradient is a fast algorithm which belongs to a class of second-order optimization method. The results verify this as almost the same accuracy is obtained by CSCG in less number of epochs.

3.2.3 Training Parameters:

Learning Rate:

LEARNING RATE	HIDDEN LAYER	TEST ACCURACY	VALIDATION ACCURACY	TRAINING ACCURACY
0.001	13	83.781	80.885	91.195
0.01		87.41	85.519	94.275
0.5		71.934	76.213	78.108
0.001	10	80.486	80.219	90.866
0.01		87.404	85.514	95.158
0.5		76.268	72.929	73.806

Cleveland Graddesc

Conclusion: If the learning rate is too high(0.5) or too small (0.001) the network response is poor. With learning rate = {0.5,0.9}, network becomes unstable.
It appears that learning rate of 0.01 is an optimal value in this case.

Momentum:

LEARNING RATE	MOMENTUM	HIDDEN LAYER	TEST ACCURACY	VALIDATION ACCURACY	TRAINING ACCURACY
0.01	0.001	5	82.792	79.913	91.418
	0.01		83.12	81.557	92.517
	0.05		82.12	79.913	91.969
	0.1		82.765	81.863	90.427
	0.5		83.454	78.585	90.54
	0.99		82.137	78.557	88.561

Conclusion: If the momentum is too high(0.99) the network response is poor. With momentum = 0.9, network becomes unstable.
It appears that momentum of 0.5 is an optimal value in this case.

It was also seen that for learning rate ->0, the use of momentum ->1 produces increasing speed convergence and for learning rate ->1, the momentum -> 0 is required to ensure learning stability.

3.2.4 Weight Regularization:

LEARNING RATE	MOMENTUM	PRIOR	HIDDEN LAYER	TEST ACCURACY	VALIDATION ACCURACY	TRAINING ACCURACY
0.01	0.05	0.01	6	86.082	80.251	92.298
		0.1		83.443	79.907	92.296
		0.9		82.137	82.546	90.756
	0.9	0	6	79.814	83.552	88.783
	0.9	0.9	6	82.475	82.557	91.307
	0.9	0.1	6	84.12	80.858	92.191
0.9	0.9	0.1	30	73.945	75.23	74.364
0.9	0.9	0.9	30	61.672	61.421	63.37

Cleveland Graddesc

Conclusion: Weight regularization is done to penalize the weights in order to eliminate network overfitting. From the table above it is verified that a very high value of prior will result in the accuracy being hit badly. Also, if momentum is set to very high value and prior is set to an appropriate value then the network actually generalizes well. If all three i.e learning rate, momentum and prior are high the network underfitts.

LIST OF REFERENCES

1. http://www.ecosystemvaluation.org/hedonic_pricing.htm
2. Introduction to Netlab http://www.ncrg.aston.ac.uk/netlab/index.php
3. Neural Networks A Comprehensive Foundation Second Edition--Simon Haykin McMaster University Hamilton, Ontario, Canada.
4. Neural Networks Metlab Help.
5. http://elsa.berkeley.edu/sst/regression.html Regression Analysis

APPENDIX A

Scatter Plots For Boston Data

The scatter plots revealed the following:

CRIM Vs MEDV: The plot suggests that there could be two subgroups. Also the correlation coefficient shows a relative strong negative relation.

ZN Vs MEDV: There is a large cluster of observations for which Zn is equal to 0. There is no clear relation.

INDUS Vs MEDV: The plot affirms the correlation coefficient which gives a strong negative relation. This could be explained by the fact that non-retail business lead to sound and air pollution. Therefore, an important variable for prediction of MEDV.

CHAS Vs MEDV: The relation is flat. CHAS value is constant.

NOX Vs MEDV: The two are negatively related. NOX, should be considered carefully as we know that the data was originally set to estimated whether pollution had an effect on pricing.

RM Vs MEDV: RM is possible measure for the size of the houses. Plot shows a cloud which is clearly upward sloping. Thus indicating a linear dependency.

AGE Vs MEDV: There is no clear connection visible.

DIS Vs MEDV: The plot hardly reveals any dependency. The table above shows positive relation.

RAD Vs MEDV: The plot suggests two subgroups.

TAX Vs MEDV: Again, the plot suggests two subgroups.

PTRATIO Vs MEDV: Indicates a negative relation i.e the less teachers there are per pupil, the less people pay on median for their dwellings.

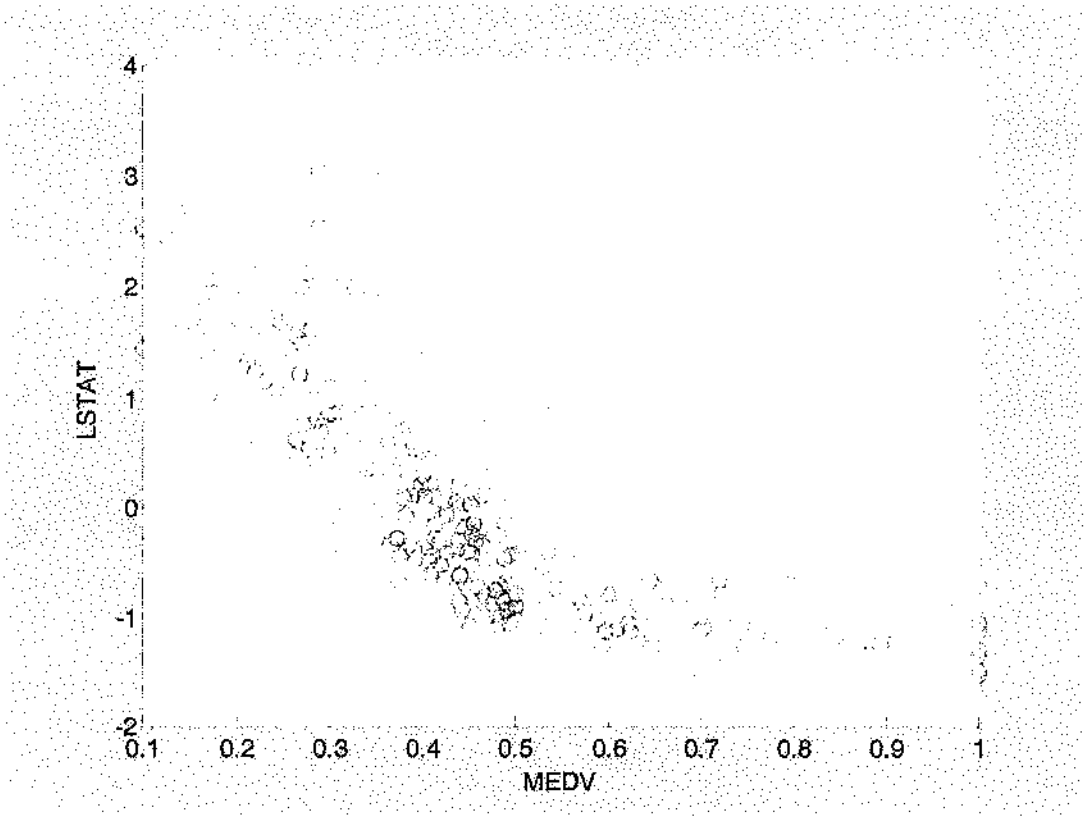
B Vs MEDV: Is positively related with MEDV.

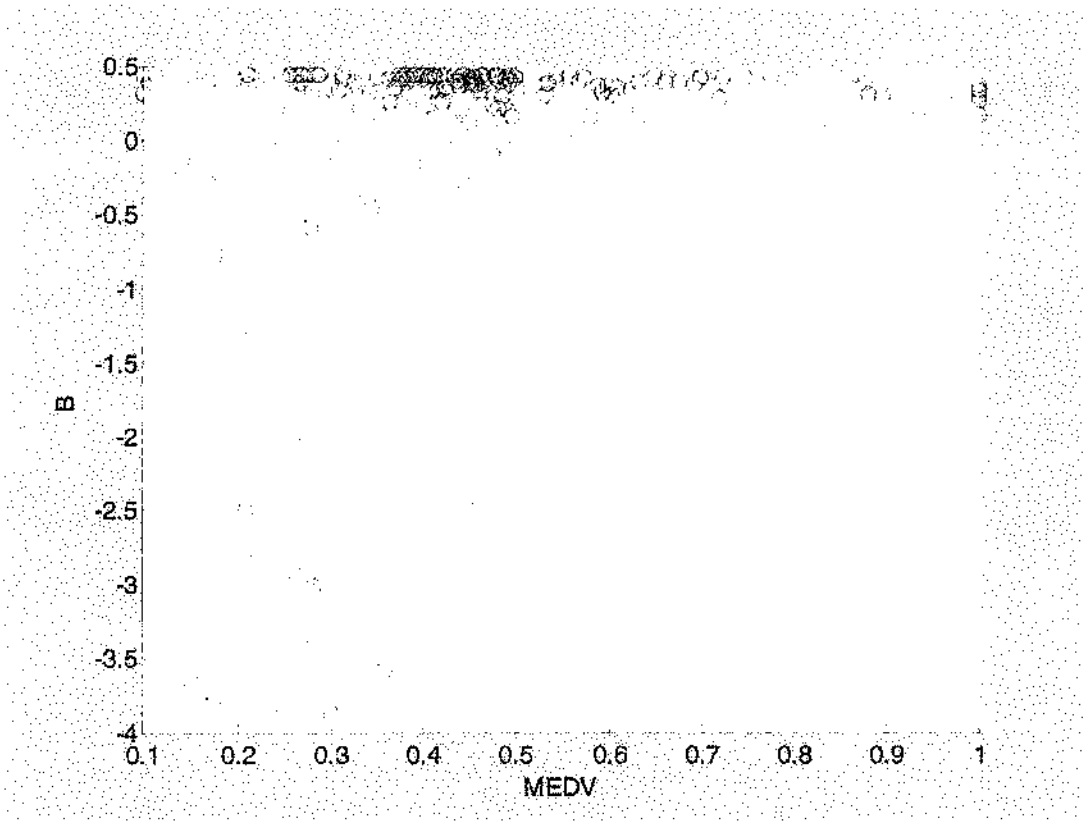
LSTAT Vs MEDV: LSTAT exhibits the clearest negative relation with MEDV.

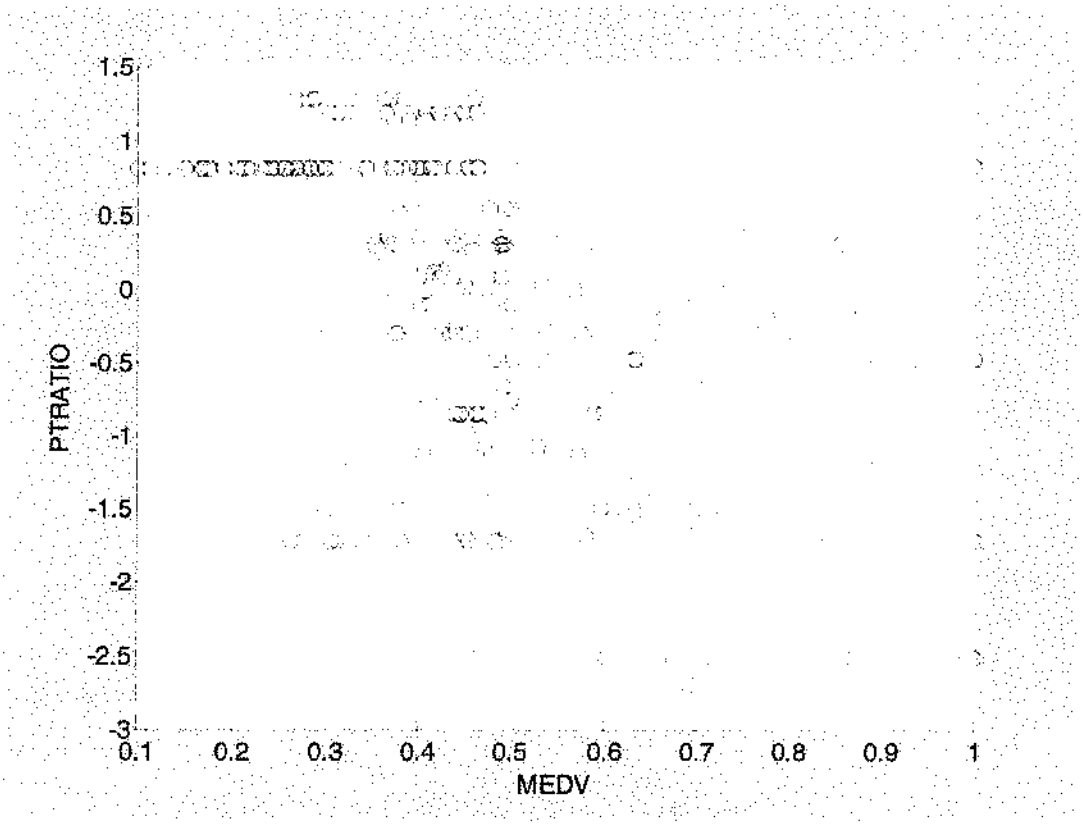
Code to plot the scatter plots in Metlab:

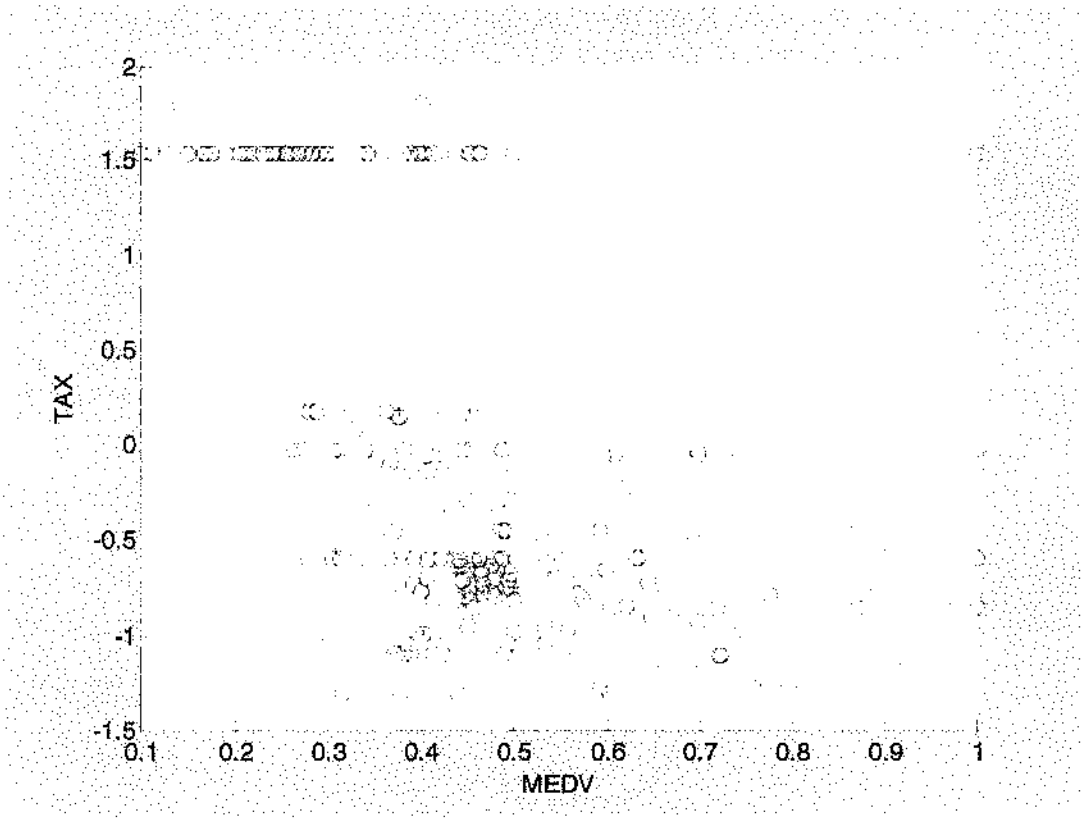
```
scatter (Boston_Training(1:253,14),Boston_Training(1:253,9));
%Legend('Training','Validation','Test', 1);
ylabel('EAD'); xlabel('MEDV')
```

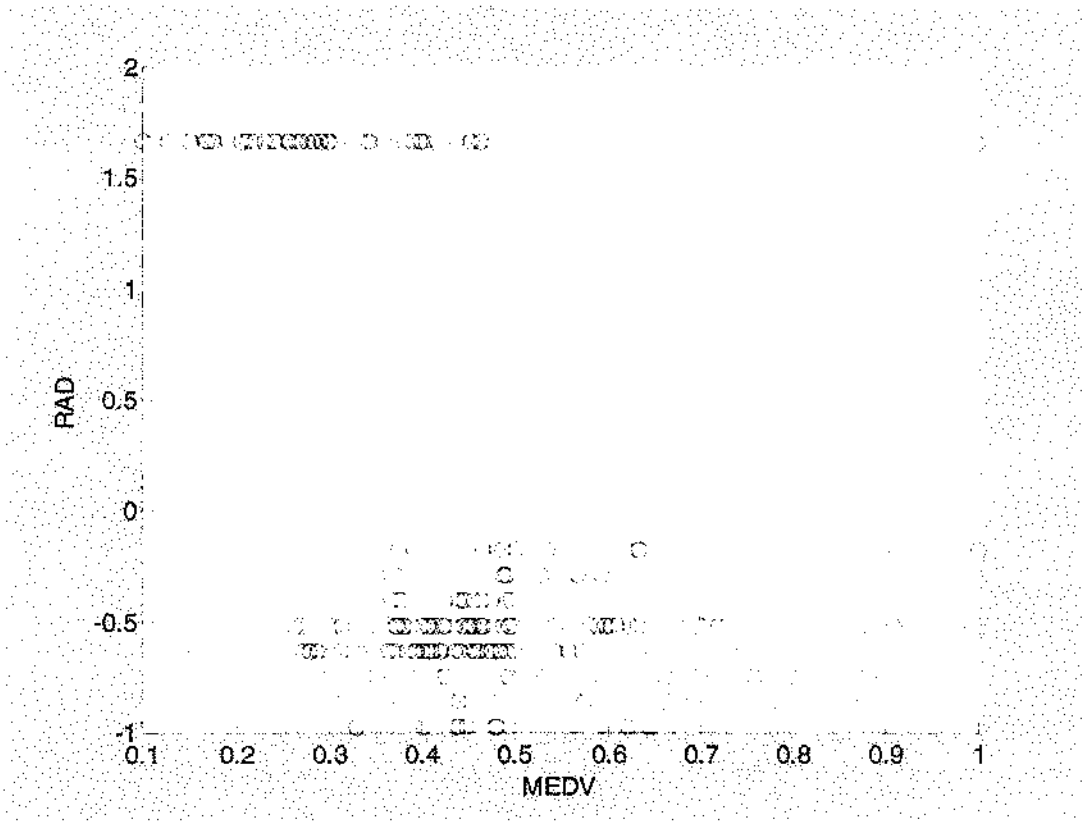
Plots:

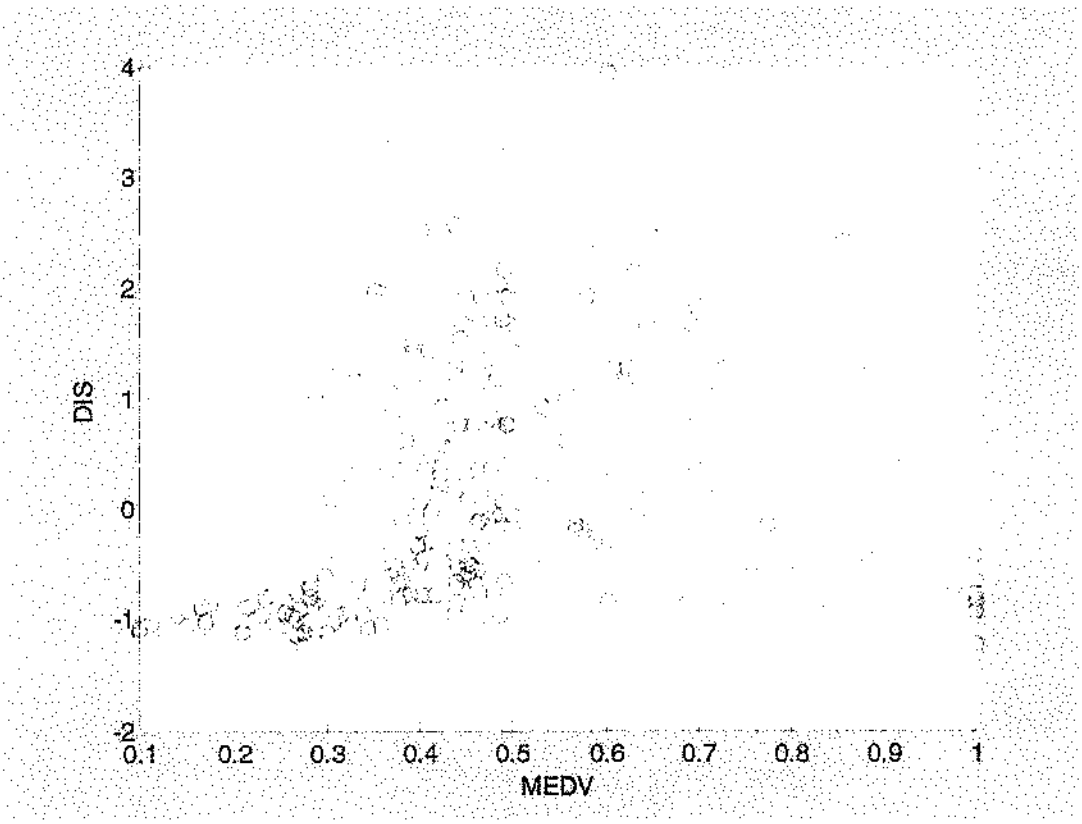


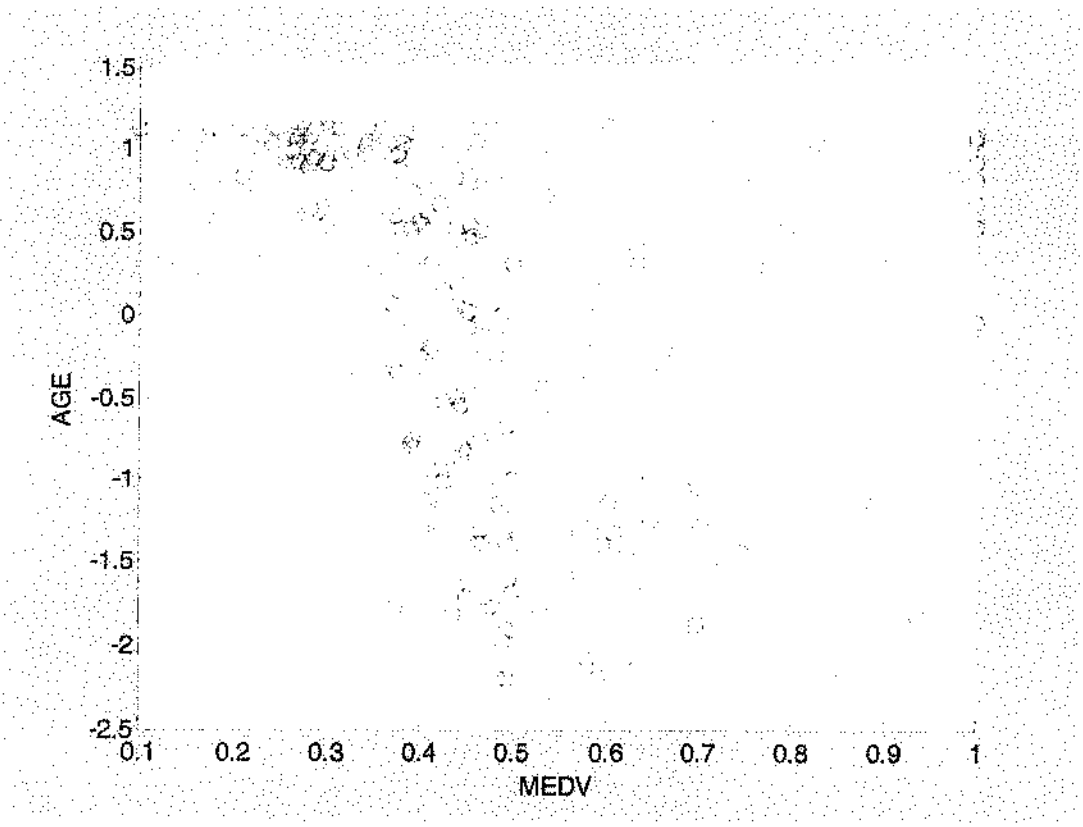

0.5
0
-0.5
-1
-1.5
B
-2
-2.5
-3
-3.5
-4
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
MEDV

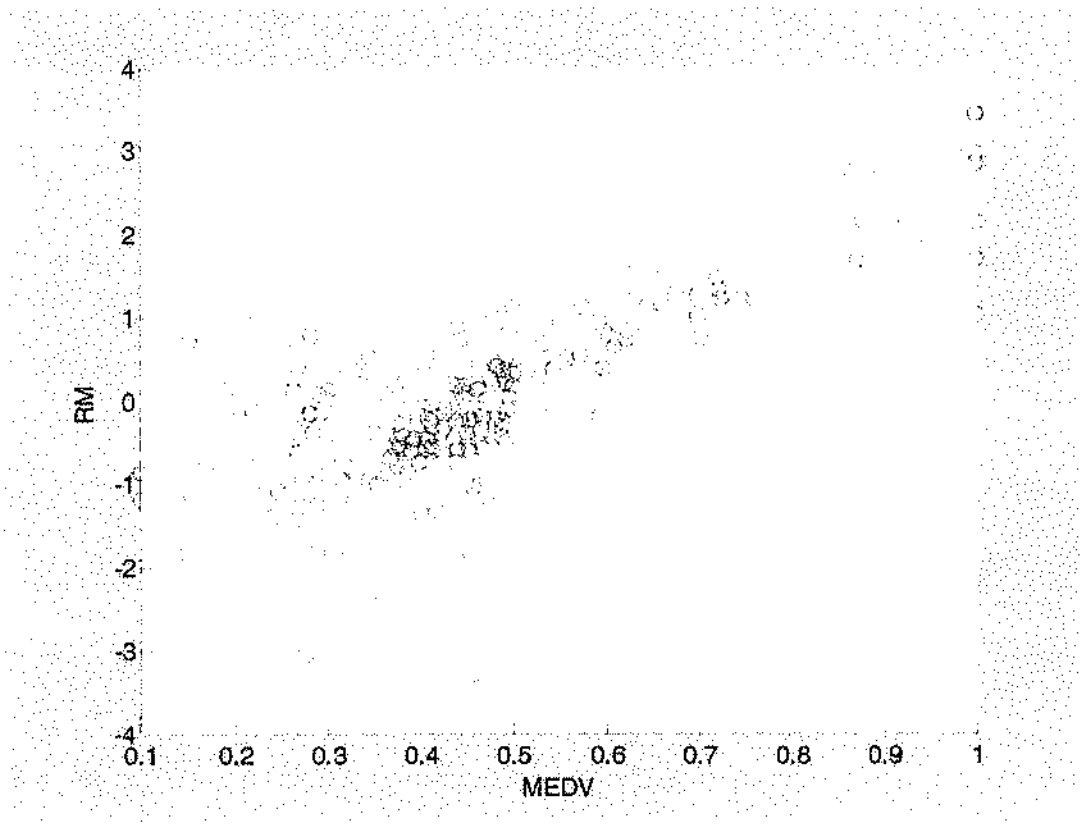

1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
-3
PTRATIO
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
MEDV

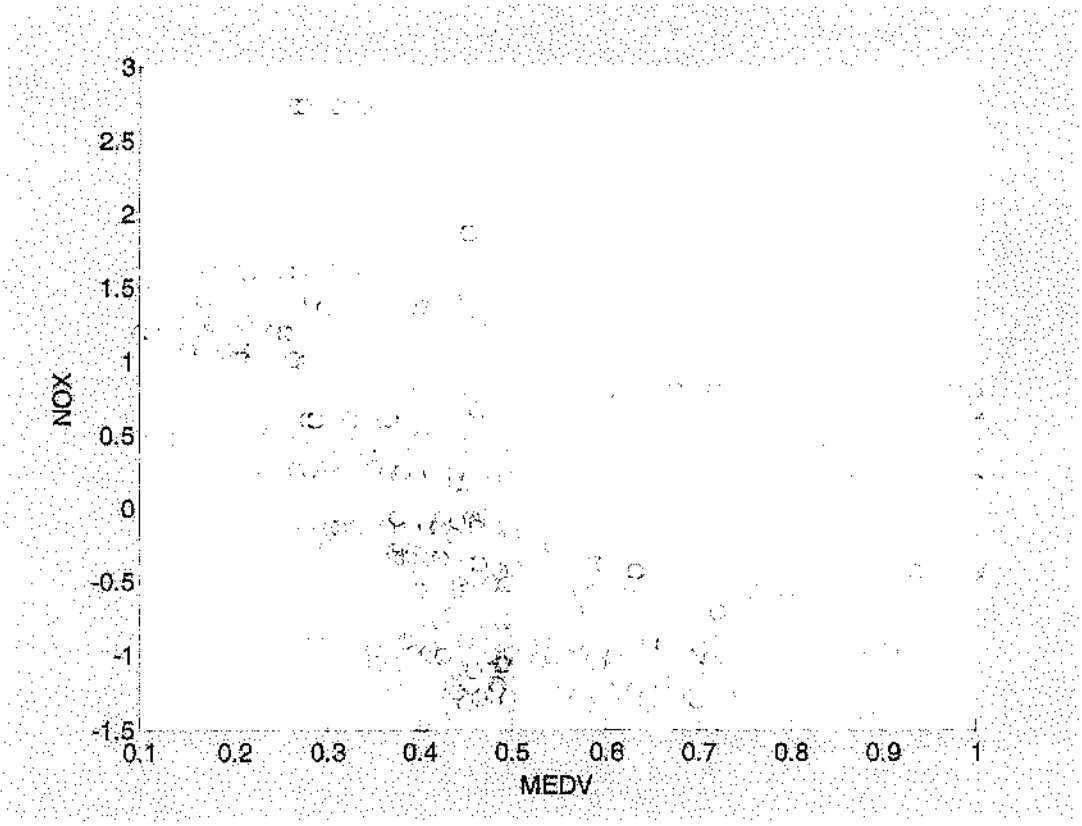

2
1.5
1
0.5
0
-0.5
-1
-1.5
TAX
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
MEDV

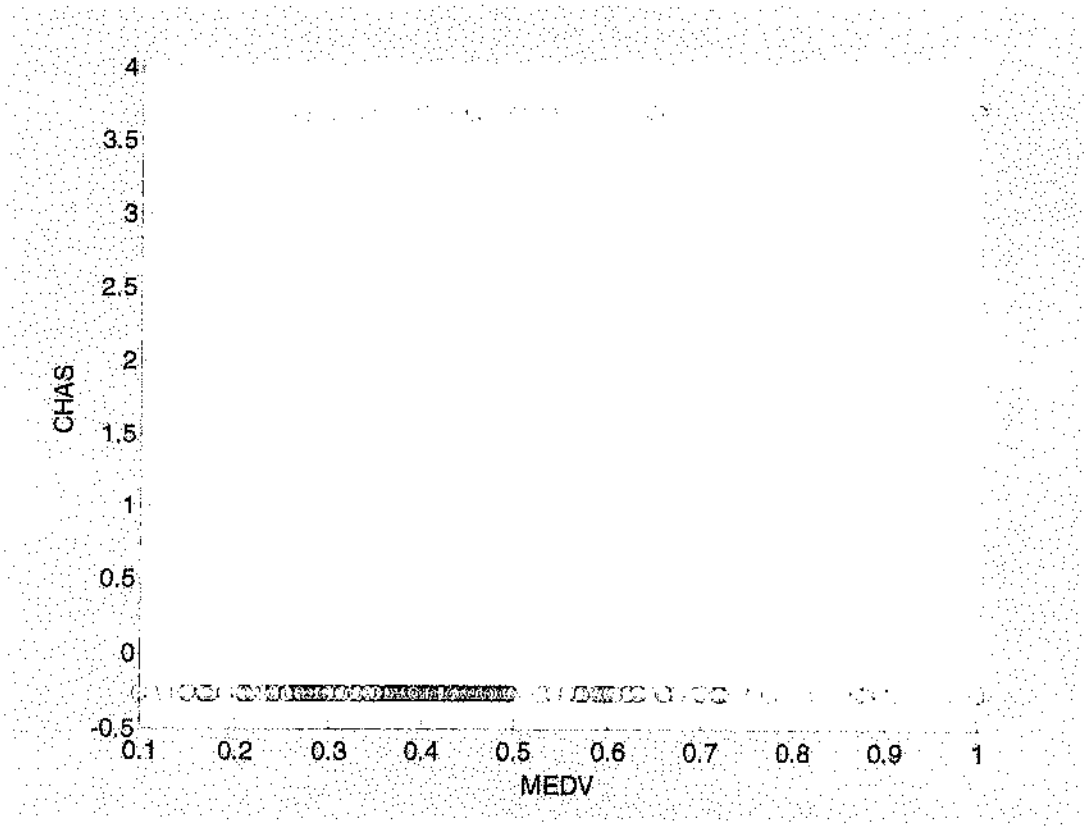

2
1.5
1
0.5
0
-0.5
-1
RAD
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
MEDV

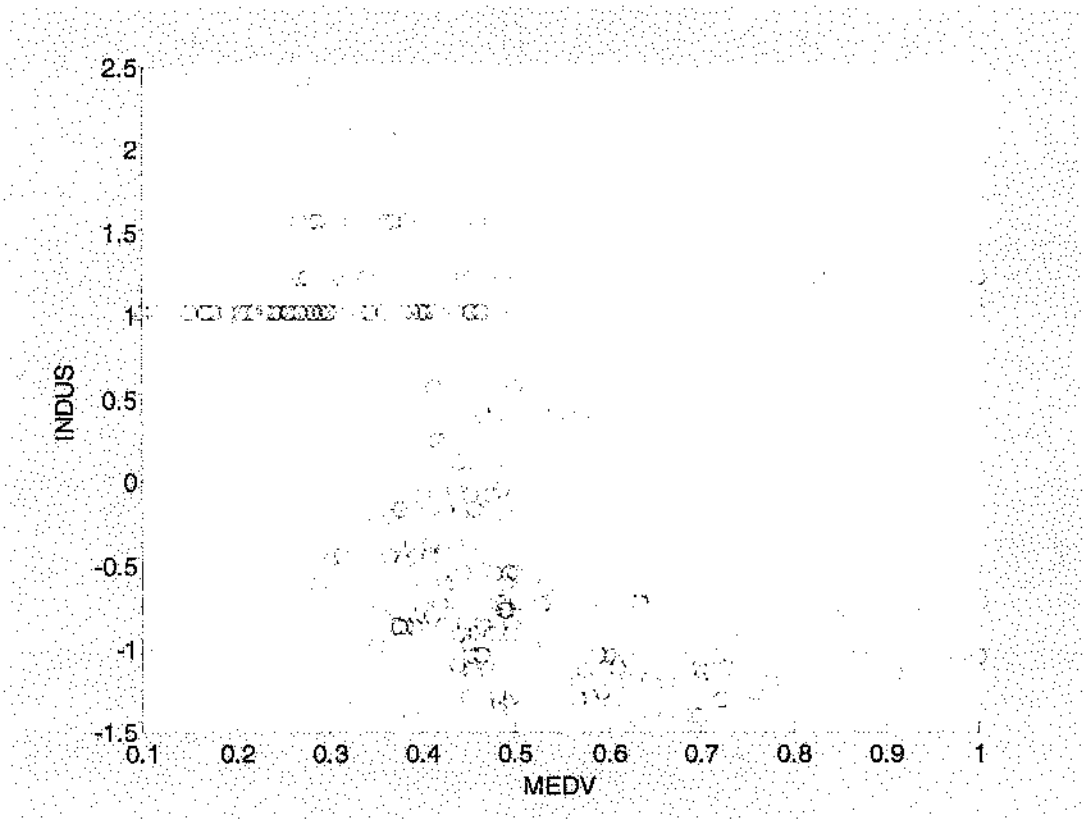

DIS
MEDV
4
3
2
1
0
-1
-2
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

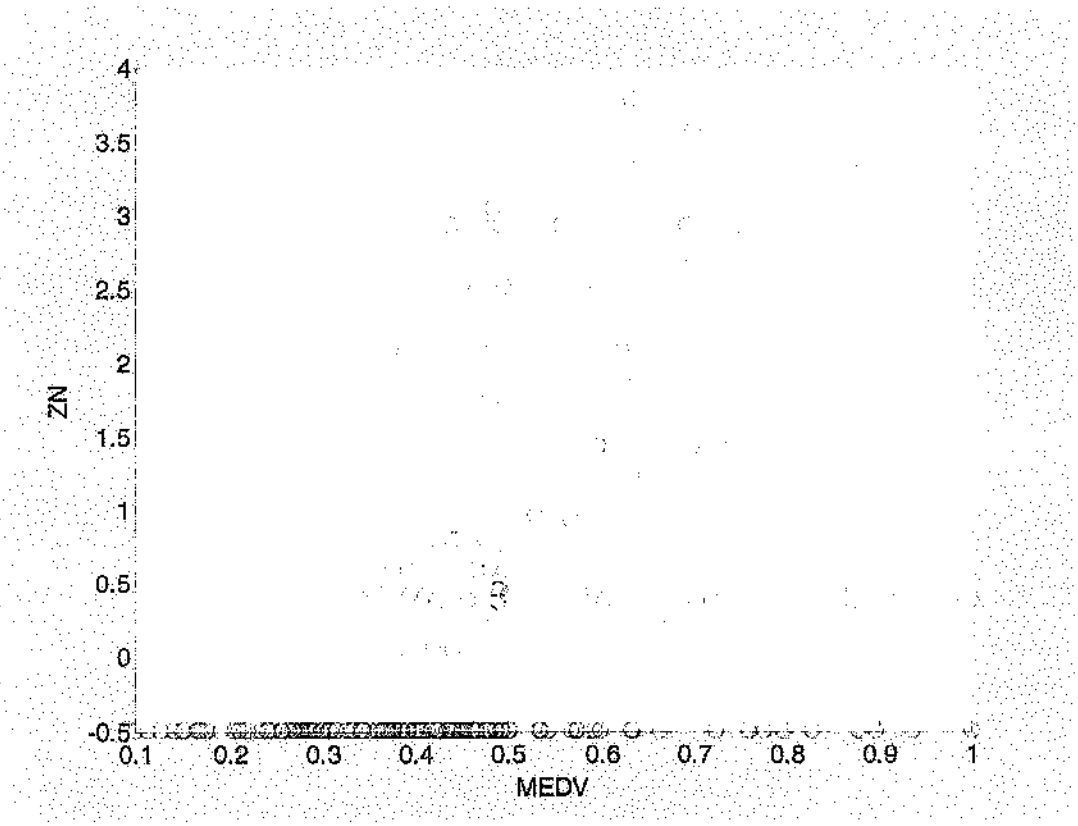

AGE
MEDV
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1

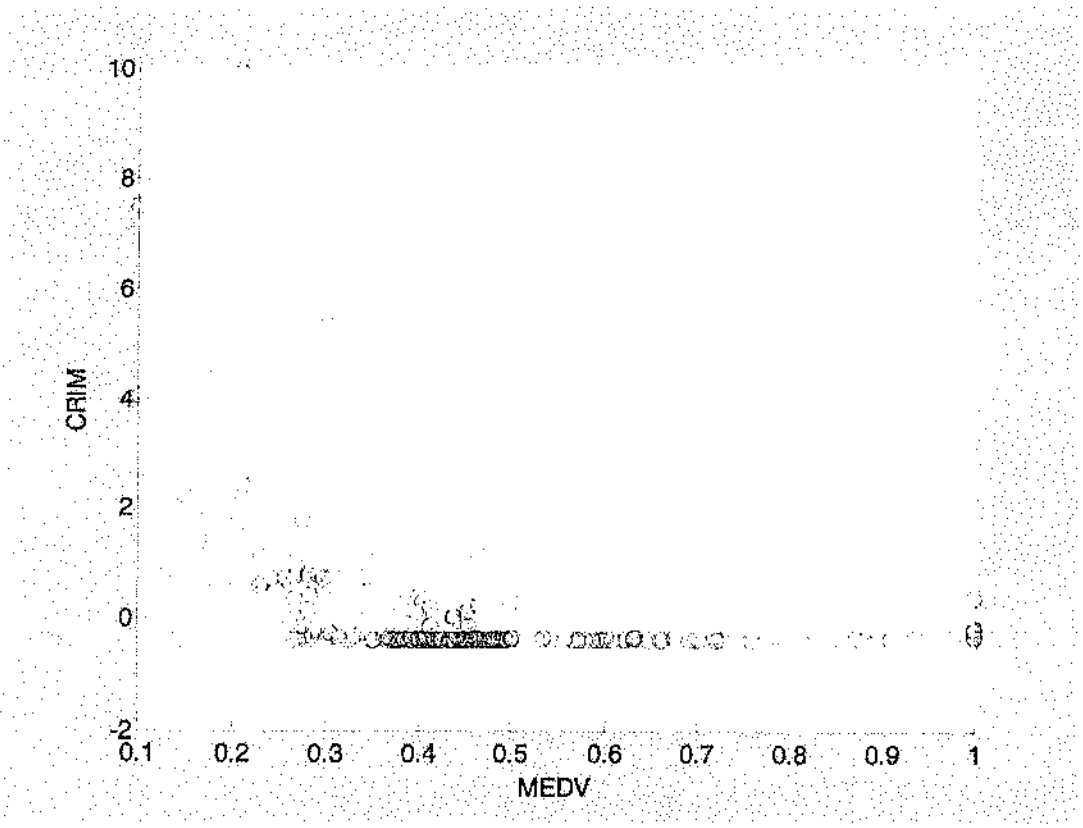

4
3
2
1
0
-1
-2
-3
-4
RM
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
MEDV


NOX
3
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
MEDV


CHAS
4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
MEDV


2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
INDUS
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
MEDV


ZN
4
3.5
3
2.5
2
1.5
1
0.5
0
-0.5
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
MEDV


CRIM
10
8
6
4
2
0
-2
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
MEDV

APPENDIX B

The Boston house-price data of Harrison, D. and Rubinfeld, D.L. 'Hedonic prices and the demand for clean air', J. Environ. Economics & Management, vol.5, 81-102, 1978.

Variables in order:

CRIM per capita crime rate by town
ZN proportion of residential land zoned for lots over 25,000 sq.ft.
INDUS proportion of non-retail business acres per town
CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise)
NOX nitric oxides concentration (parts per 10 million)
RM average number of rooms per dwelling
AGE proportion of owner-occupied units built prior to 1940
DIS weighted distances to five Boston employment centres
RAD index of accessibility to radial highways
TAX full-value property-tax rate per $10,000
PTRATIO pupil-teacher ratio by town
B 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
LSTAT % lower status of the population
MEDV Median value of owner-occupied homes in $1000's

APPENDIX C

The **Principal Component Analysis** for Boston data set was carried out. From scatter plots I had concluded that six variables account for the 14th feature. Thus during the analysis I aimed to retained those principal components for which account to *x%* of variation and returned R=6.

```
[ptrans,transMat] = prepca(pn,0.01);
[R,Q] = size(ptrans);
R = 12;
Q = 506;

[ptrans,transMat] = prepca(pn,0.02);
[R,Q] = size(ptrans);
R = 9;
Q = 506;

[ptrans,transMat] = prepca(pn,0.05);
[R,Q] = size(ptrans);
R = 6;
Q = 506;

[ptrans,transMat] = prepca(pn,0.06);
[R,Q] = size(ptrans);
R = 5;
Q = 506;


[ptrans,transMat] = prepca(pn,0.001); to(0.004)
[R,Q] = size(ptrans);
R = 13;
Q = 506
```

Thus it was decided that those principal components which account for 95% of the variation in data set be retained.