Project milestone report for CS229: Blood Pressure detection from PPG signal - Sharath Ananth (SUID: sharath2)

1. Introduction

It is known that heart rate and SpO2 oxygenation can be detected from a cell phone camera and flash light. For example the successful app “Instant Heart Rate” from Azumio uses the cell phone camera and the flash light to measure heart rate. This app has been downloaded more than 10 million times. It is also well known that SpO2 levels can be determined by using the same method.

Other research papers have connect SpO2 graph, also called Photoplethysmograph (PPG) with Blood Pressure, for example references [1-6]. The ultimate idea of this project is to develop an algorithm which will allow a cell phone app to analyze signals from the camera to estimate blood pressure. Note that only the algorithm will be developed in this project and not the app itself.

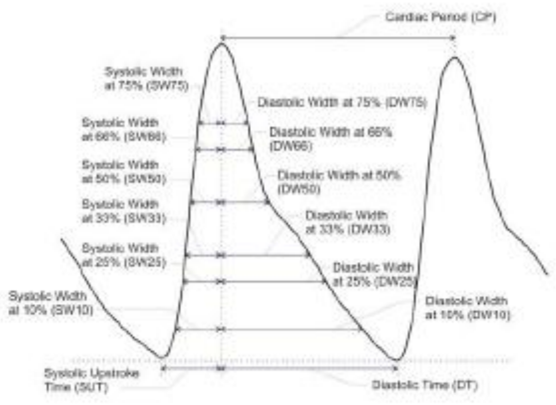
According to the American National Standards of the Association of Medical Instrumentation, the mean absolute difference between the device and the mercury standard sphygmomanometer must be less than 5mmHg, and the standard deviation must be less than 8 mmHg. Hence the goal of this project is to be within these bounds.

2. Background

2. 1 Structure of signal waveform

[4] Explains that a MIMIC data base exists which contains BP information and the corresponding PPG signal. This data base is used in this project. The aim then is to extract features from the raw PPG signal and use this as input to a Neural Network.

The figures below explains the structure of the PPG signal and information extracted from this signal.



Cardiac Period (CP)
Systolic Width at 75% (SW75)
Diastolic Width at 75% (DW75)
Systolic Width at 66% (SW66)
Diastolic Width at 66% (DW66)
Systolic Width at 50% (SW50)
Diastolic Width at 50% (DW50)
Systolic Width at 33% (SW33)
Diastolic Width at 33% (DW33)
Systolic Width at 25% (SW25)
Diastolic Width at 25% (DW25)
Systolic Width at 10% (SW10)
Diastolic Width at 10% (DW10)
Systolic Upstroke Time (SUT)
Diastolic Time (DT)

Figure 1 Structure of the PPG signal and the parameters extracted from them in [4]

The underlying intuition behind this extraction comes from [1-4] which show that there is a linear correlation between the blood pressure (BP) and heart beat duration as computed from the PPG signal (Cardiac Period). In [1-2] the systolic upstroke time, diastolic time as well as the width of 2/3 and ½ pulse amplitude were considered as the possible parameters and the diastolic time was stated as the more correlated to the BP. Tests show that the higher the BP the shorter is the duration of every heart beat. However, more tests with different signals show that such correlation is not always linear. [4] states that different authors provide different coefficients for estimating the BP from the CP, but such coefficients are dependent on the person and need adjustment for each person.

The aim of [4] and the first part of this project is to use a Neural Network (NN) to estimate the Systolic and Diastolic blood pressure using a NN and the features described above.

2.2 Machine learning algorithms

Two machine learning algorithms were evaluated in this report. One is an incremental gradient descent and the other is a Neural Network. These two methods are discussed further in this section

2.2.1 Incremental gradient descent

Incremental gradient descent is a variation of the batch gradient descent algorithm where the parameters are updated for every training example. In batch gradient descent the parameters are updated once after looking at all the samples. Incremental gradient descent was chosen since the data set generated is quite large and it is expected that the algorithm is able to converge without having to use all the samples.

It is assumed that the output 'y' is a linear combination of the input 'x' in the following fashion:

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 = h_\theta(x)$$

Here $\vec{x}$ is the input vector, $\vec{y}$ is the output vector and $\vec{\theta}$ is the vector being estimated by incremental gradient descent.

The algorithm is as follows:

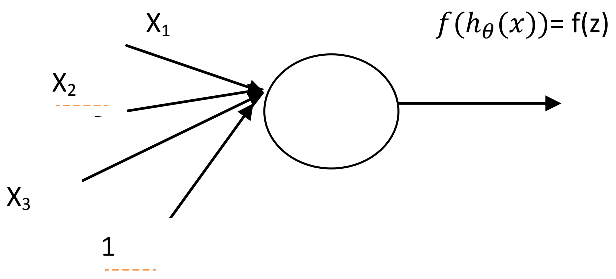
```
Loop {
    For i = 1 to number_of_input
        $\theta_j := \theta_j + \alpha(y^i - h_\theta(x^i))x_j^i$
     }
```

The cost function $J(\theta) = (\frac{1}{2m}) \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2$ was evaluated to measure convergence.

2.2.2 Neural Network

The other algorithm evaluated was the classic back propagation Neural Network. Matlab Neural Network toolbox implements a Levenberg-Marquardt backpropagation algorithm for the training. This algorithm is described in [10] and the Matlab help page. The basic Neural Network algorithm with error back-propagation is first described here following the material in [11]

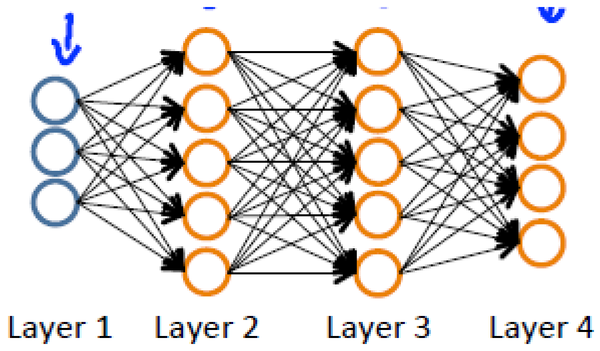
Neural network provides a method of defining a complex non-linear form of hypothesis to fit to the data $((\vec{x}, \vec{y}) using\ the\ parameter\ \vec{\theta}$. Consider the simplest Neural Network with one Neuron as shown below:



The "neuron" is a computation unit which takes as input $\vec{x}$ and outputs $f(\theta_0 + \theta_1 x_1 + \theta_2 x_2) = f(h_\theta(x))$ = f(z)

That is the single neuron is very similar to the input-output mapping used in incremental gradient descent. The weight vector is equivalent to $parameter\ \vec{\theta}$. The function f() is the activation function. So the individual inputs are X1-X3 are weighed by $\theta$and summed up. A function f() is applied this sum to generate the output.

A neural network is put together by hooking together many of our simple neurons so that the output of a neuron is the input of another. An example of a 4 layer NN is below (figure from [11]), where the first layer is the input layer and the 4th layer is the output layer. The middle two layers are the hidden layers.



To simplify notation the output of each neuron is denoted by a_{node}^{layer}. That just means that f($h_\theta(x)$) is denoted by 'a_{node}^{layer}'. Alternatively f(z) = a_{node}^{layer}.

For example in the figure above the input's are layer 1. So the first output from layer 1, is a_1^1 which is simply the first input x0, the next output from layer 1 is denoted as a_2^1 which is simply the next input x1.

Error Back-propagation Algorithm:

The back-propagation algorithm is a method of finding the weights of the neurons for all the layers so that a cost function is minimized. The cost function used in NN is very similar to the cost function described in incremental gradient descent with the addition of a regularization parameter. The new cost function is $J(\theta) = \left(\frac{1}{2m}\right) \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2 + \gamma/2 \sum_{i,j,l} \theta_{i,j}^l$ where l=layer, (i,j) = from i'th node to j'th node.

As in most cost functions the update function is as follows:

$$\theta_{i,j}^l := \theta_{i,j}^l - \alpha \frac{\partial}{\partial \theta_{ij}^l} J(\theta)$$

Computing the partial derivatives above is the key step which back-propagation helps in. Back-propagation gives an efficient way to compute these partial derivatives

As explained in [11] the intuition behind back-propagation is as follows: given every training example (x,y) the neural network is run in a 'forward pass' fashion to compute all activations throughout the network, including the output value $h_\theta(x)$. Then for each node in each layer, the algorithm would like to compute the "error term" δ_i^l that measures how much the node was "responsible" for any errors in the output. For the output node, one can directly measure the difference between that node's activation and the true target value to compute the δ_i^l. For the hidden units, the error term is computed as a weighted average of the error terms of the nodes. That is:

(i) Perform feedforward pass, computing the activation for layers 2, 3 and so on upto the output layer

(ii) For the output layer, the error is $(h_\theta(x^i) - y^i)^2$ and the following is set:

$$\delta_i^{last\_layer} = \frac{\partial}{\partial z_i^{lastlayer}} \left(\frac{1}{2}\right) ||h_\theta(x^i) - y^i||^2 = -(y_i - a_i^{lastlayer}).f'(z_i^{lastlayer})$$

(iii) For each layer (going backwards from the last layer), set

$$\delta_i^l = \left(\sum_{j=1}^{s_l} \theta_{ij}^l \, \delta_j^{(l+1)} \right) f'(z_i^l)$$

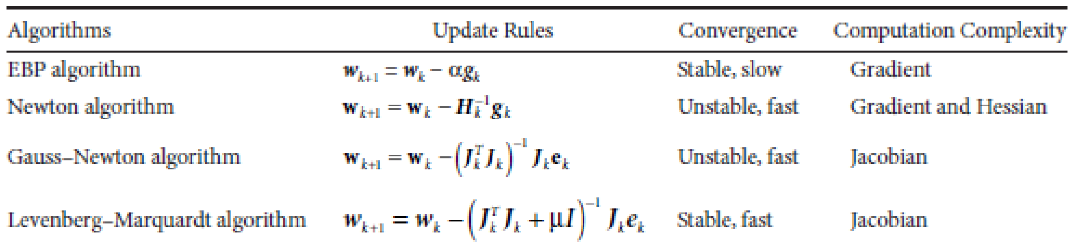
(iv) The partial derivative term is now set as

$$\frac{\partial}{\partial \theta_{ij}^l} J(\theta) = a_j^l \delta_i^{(l+1)}$$

Levenberg-Marquardt back-propagation

[12] gives a good overview of the Levenberg-Marquardt algorithm. The primary difference is how the weight functions are updated and the overview from[12] is shown below

TABLE 12.1 Specifications of Different Algorithms



Algorithms	Update Rules	Convergence	Computation Complexity
EBP algorithm	$w_{k+1} = w_k - \alpha g_k$	Stable, slow	Gradient
Newton algorithm	$\mathbf{w}_{k+1} = \mathbf{w}_k - H_k^{-1} \mathbf{g}_k$	Unstable, fast	Gradient and Hessian
Gauss–Newton algorithm	$\mathbf{w}_{k+1} = \mathbf{w}_k - \left(J_k^T J_k\right)^{-1} J_k \mathbf{e}_k$	Unstable, fast	Jacobian
Levenberg–Marquardt algorithm	$w_{k+1} = w_k - \left(J_k^T J_k + \mu I\right)^{-1} J_k e_k$	Stable, fast	Jacobian

The principal ideal behind the Gauss-Newton algorithm is the approximation of the Hessian Matrix by the product of two Jacobians. That is $H_k^{-1} \approx (J_k^T J_k)^{-1}$

The principal idea behind the Levenberg-Marquardt algorithm is the approximation of the Jacobin matrix as the sum of two elements. That is $H_k^{-1} \approx (J_k^T J_k + \mu I)^{-1}$

With $\mu = 0$ the algorithm is using Gauss-Newton and with a large value of μ the algorithm is using the steepest descent. The algorithm switches the value of μ during training. Initially the algorithm uses a small value so that the algorithm is approximating the Gauss-Newton, and later it switches to a large value of μ and the algorithm switches to the Error Back propagation. This allows the algorithm to be able to get the benefit of fast convergence of Gauss-Newton and the stability of Error Back Propagain.

The rest of the changes to the algorithm are to enable it to implement the Gauss-Newton mechanism rather than the EBP. The steps are as follows:

(i) Compute the Feedforward just as in the EBP algorithm and compute the error.
(ii) Compute the Jacobian (by using the chain rule)
(iii) Compute the error gradient = J^{T*} Error
(iv) Compute the Hessian inverse as $H_k^{-1} \approx (J_k^T J_k + \mu I)^{-1}$ and update the new weights
(v) Repeat above till converge

Luckily Matlab's tool-box has already implemented this algorithm!

3. Pre-processing step

Considerable effort is focused on the pre-processing step which allows for extraction of the parameters described above from a PPG signal and this is a non-trival task. The data in the MIMIC database is noisy and clipped. For the signals which are not noisy and clipped the data does not fully follow the structure shown in Figure 1. A typical PPG signal is shown in figure below:

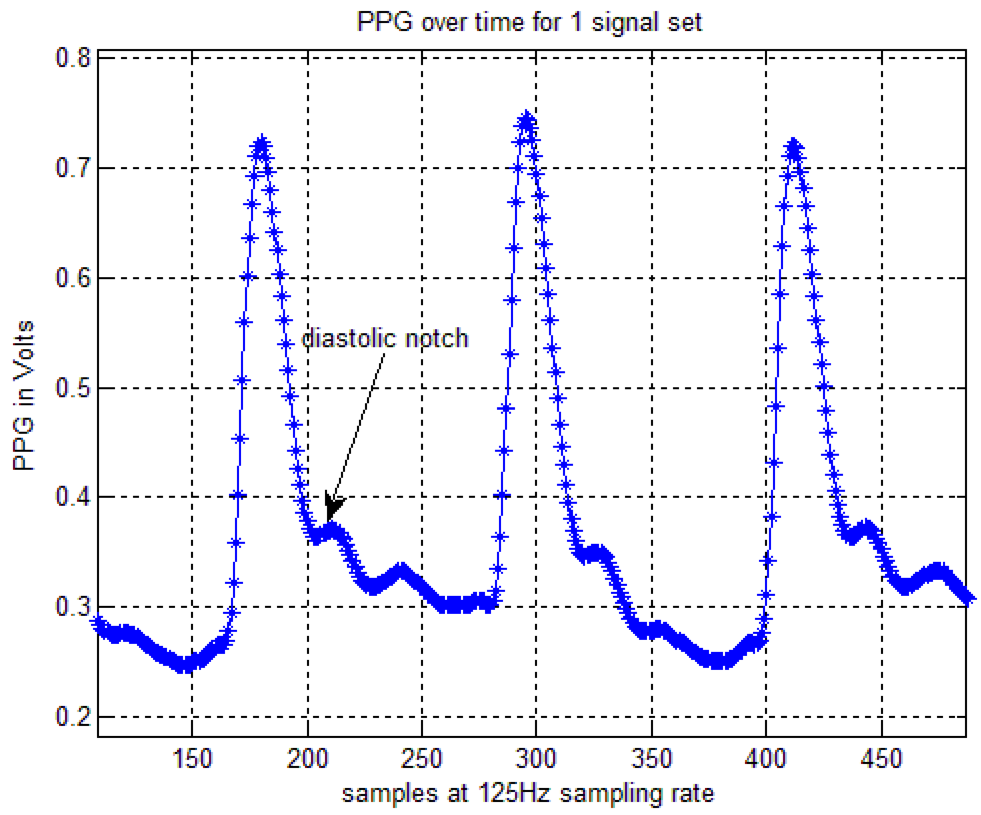


Figure 2 RAW PPG signal

As is seen in the above figure there is a peculiar shape to the PPG signal. This shape repeated in a few good waveforms that it was required to understand this some more. [1] Mentions that the PPG signal may be noisy such that the 'foot' or 'trough' of the signal may be hard to find. [1] uses a wavelet transform to eliminate these errors in the transform domain. [8] explains the structure of the signal nicely.

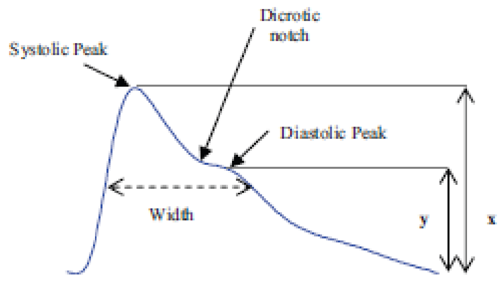


Fig. (9). A typical waveform of the PPG and its characteristic parameters, whereas the amplitude of the systolic peaks is x while y is the amplitude of the diastolic peak.

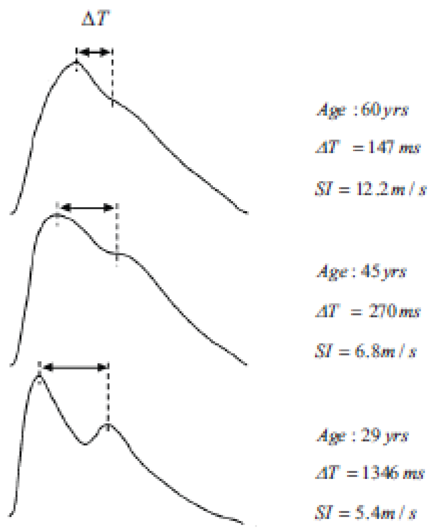


Fig. (13). Typical PPG waveforms show the parameters changes with age [62].

Figure 3 Structure of PPG signal as per [8]. It shows how the signal changes with age.

Figure 3 show's how the shape of the PPG signal can change with age. This explains why some of the recorded PPG signals show this shape and why others don't.

The challenge now is to find the terms CP, SUT, DT from the PPG signal in the presence of the Diastolic peak. The pre-processing step that is currently implemented first finds the peaks above a certain threshold. (This threshold is chosen manually for each PPG signal). Once the systolic peaks have been found the minimum value between every two Systolic peaks is the point used to find the Diastolic time (DT), the Systolic Update (SUT) and the Cardiac Periodic (CP).

4. Neural Network training and results

Matlab's neural network tool box allows one to train, validate and test the data. Currently a NN with one hidden layer and 30 neurons was used to test. The data set consisted of 30000 records generated as explained above. The test data is generated using 6 patient data, each over approximately 2 hour periods. Matlab's model splits the data set into 70% for training, 15% for validation and 15% for testing. The following graph shows the histogram of the Dystolic and Systolic Blood pressure

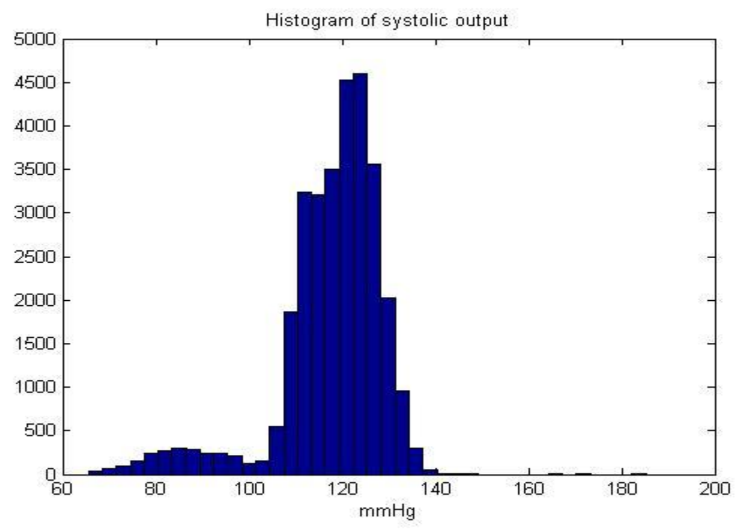


Figure 4 Systolic Blood pressure, output of Neural network

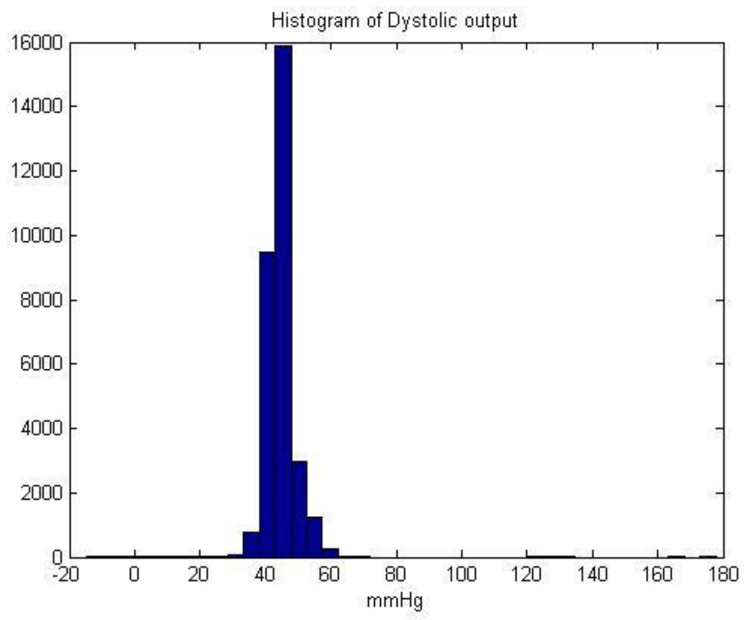
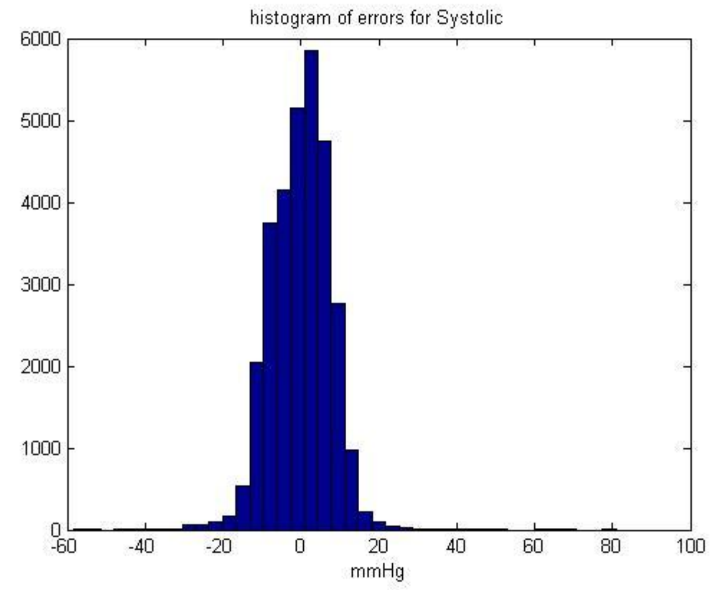


Figure 5 Dystolic blood pressure output

The error is -0.0291±7.6 mmHg and -0.0276±4.4 mmHg and the MMSE performance metric was 38.The histogram of errors is shown below



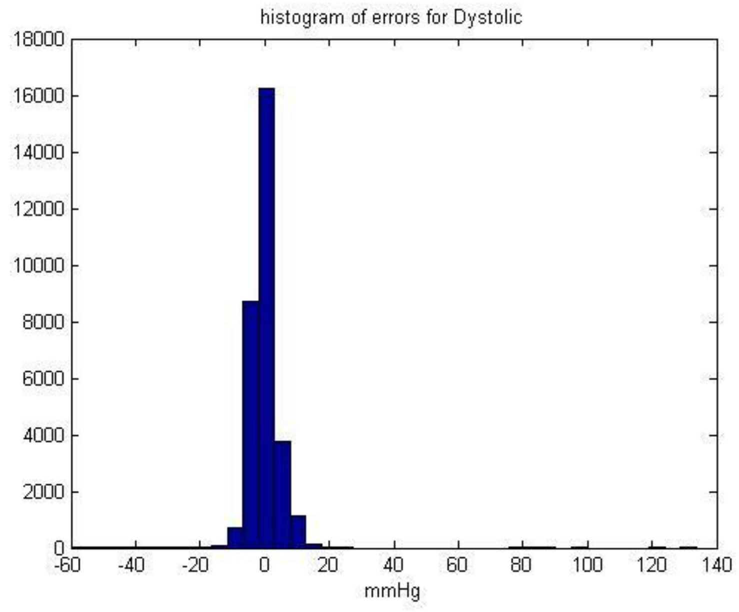


Figure 6 Histogram of errors

5. Insight and Discussion

From the first set of results it seems that the mean error is low and the variance is high. So clearly we need more parameters to be fed into the neural network. Let us first evaluate what the errors are with using 1 to 3 parameters. The table below compares the results

Parameters	Mean/std of error	Performance	Comments
CP, SUT, DT	-0.0291±7.6 mmHg and -0.0276±4.4 mmHg	38	
SUT, DT	0.051±8.2 mmHg and -0.019±4.4 mmHg	42	
CP, SUT	0.039±7.6 mmHg and -0.0096±4.33 mmHg	38.14	
CP, DT	0.03±7.3 mmHg and -0.002±4.3 mmHg	36.2	
CP	0.0212±7.54 mmHg and -0.014±4.34 mmHg	37.6	
DT	-0.0339±7.9794 mmHg -0.0245±4.3540 mmHg	41.2	
SUT	.04+9.53 mmHg .046+5.2 mmHg	59.1	

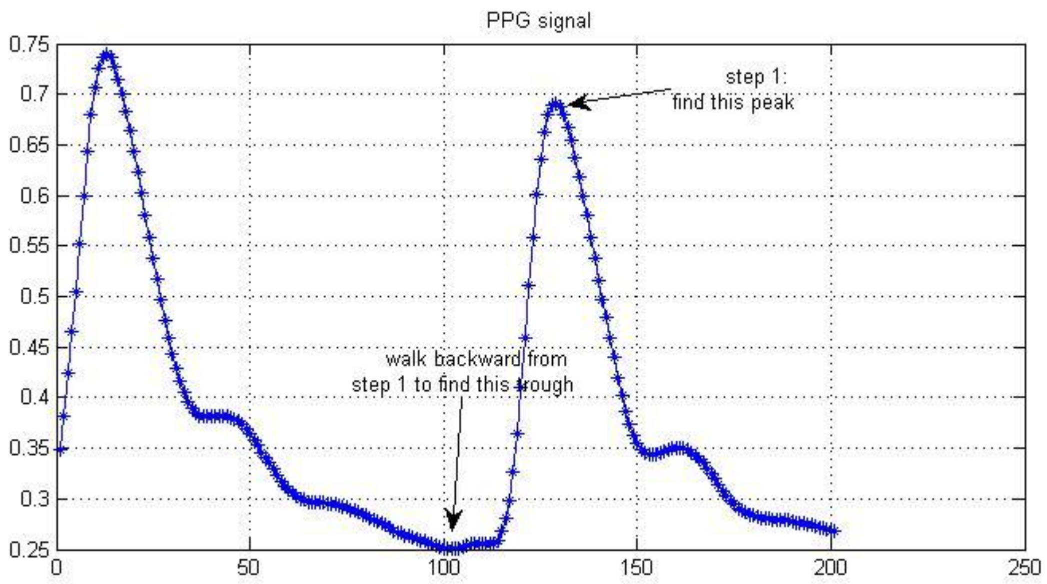
Negligible, difference between changing the different parameters. On closely examining the output from the NN and the desired output it was found that the error is dominated by impulse noise in the input-output vectors. This implied that further pre-processing is required.

6. Modified Pre-processing and results:

After further analyzing the input waveform and the pre-processing step it was seen that the peak and trough detection was finding many false peaks and troughs. This was eliminated by the following pre-processing changes

(i) The peaks cannot be very close to each other. A threshold was experimentally determined such that two peaks closer than the threshold are considered part of the same signal. This threshold is dependent on understanding how close the heart beats are. For example children will have closer thresholds than adults. Currently the threshold used is only for adults and further processing needs to be done to handle children

(ii) The troughs were found by 'walking' down the peak to find the position where the signal changes shape. This is explained in the figure below



(iii) Median filter the inputs and outputs to eliminate impulse noise
(iv) Visually inspecting the input and output waveforms to eliminate regions of severe clipping or zero samples due to instrumentation error.

The new results with these modified input and output is shown in the table below

Parameters	Mean/std of error	Performance with 30 hidden neurons	Performance with 100 hidden neurons

CP, SUT, DT	0.0016±4.43 mmHg
	-0.0083±3.64
		R = 0.995
		MSE = 12.63
			MSE = 12.35 (too small an improvement to proceed down this path)

SUT, DT	0.0062±4.45 mmHg
	-0.027±2.41 mmHg
		MSE = 13.12
		R = 0.995	

CP, SUT	0.012±4.5 mmHg
	0.0123±2.38mmHg
		MSE = 13.35
		R = 0.995	

CP, DT	-0.31±4.82 and
	-0.1024±2.4 mmHg
		MSE =14.61
		R = 0.994	

CP	0.0098±4.62 mmHg
	0.0115±2.37 mmHg
		MSE = 13.47
		R = 0.995	

DT	.02+4.69 mmHg
	.025+2.8 mmHg
		MSE = 15.99
		R = 0.994	

SUT	.03+4.96 mmHg
	.0076+2.61 mmHg
		MSE = 15.72
		R = 0.994	

Clearly there is some performance improvement by adding the additional features, but not enough to add a many more than 3. The regression plot looks as follows

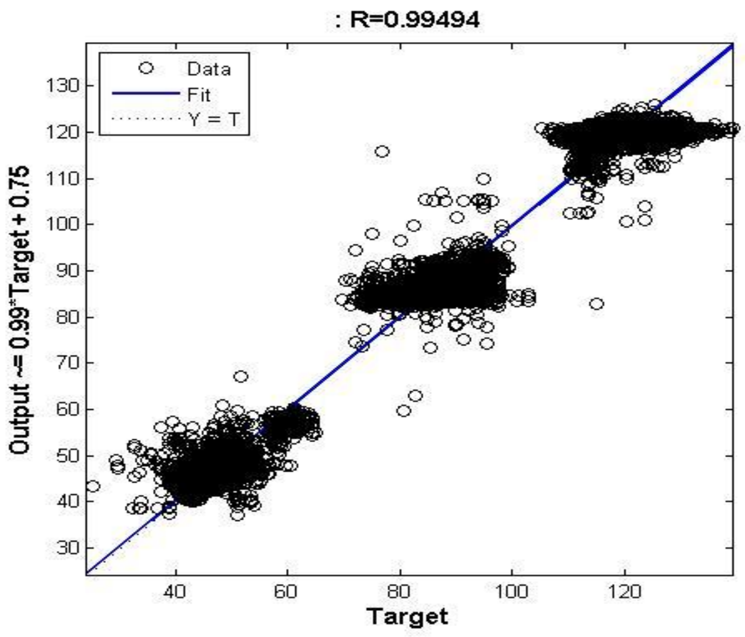


Figure 7 Regression plot using 3 inputs for 2 outputs, 30 element NN

As seen in the regression plot above, there is a reasonably good. The next section evaluates incremental gradient descent for this problem

7. Incremental gradient descent

Incremental gradient descent (as explained in 2.2.1) was also implemented and the following convergence curves were obtained using 3 inputs (SUT, DT, CP) and 2 outputs.

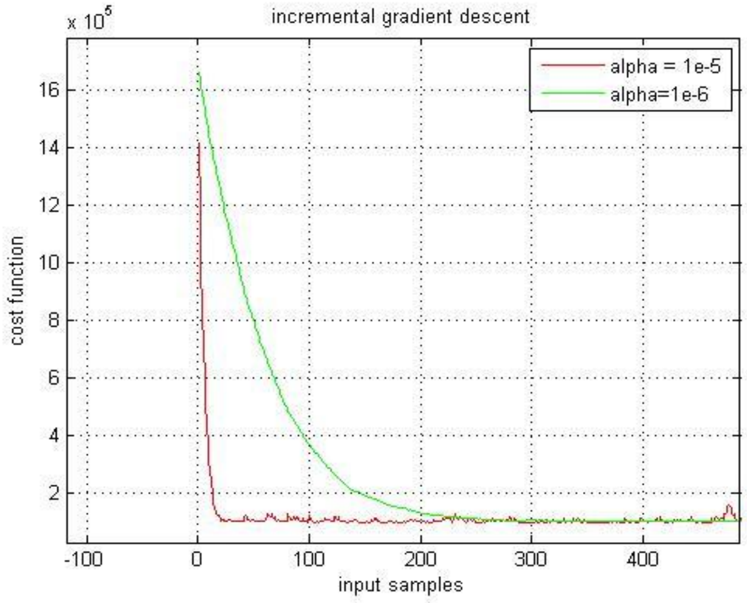


Figure 8 Cost function vs iterations for incremental gradient descent

The MMSE error was computed as ~48 (as compared to ~13 in NN). The mean square error is 5 times worse than NN. The regression plot looks as below:

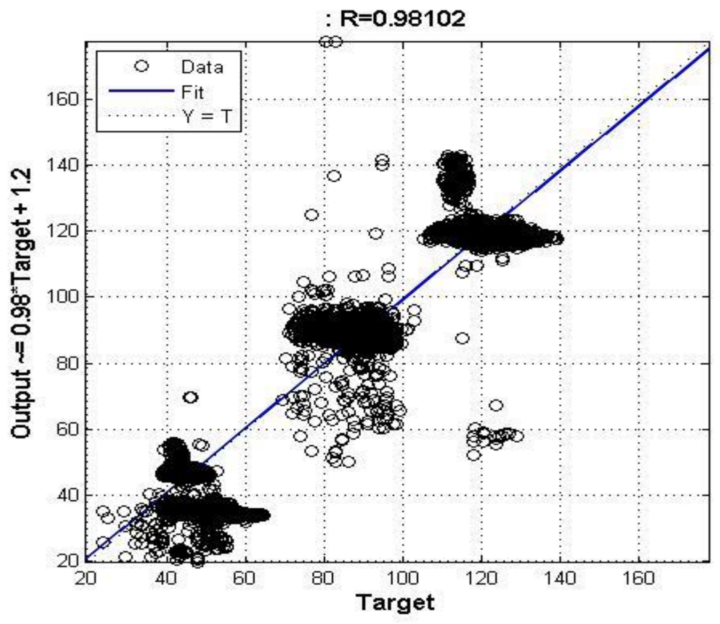


Figure 9 Regression plot using 3 inputs, using incremental gradient descent

We see that there is a good fit here as well, just not as good as what is found in the Neural Network. The table of results are below

Parameters	Mean/std of error	Comments

CP, SUT, DT	-0.08±7.03 mmHg
	1.44±6.72 mmHg
		MSE=48.3
		R = 0.9817

SUT, DT	0.69±7.06 mmHg
	1.22±7.69 mmHg
		MSE= 55
		R = 0.9792

CP, SUT	-0.1±8.2 mmHg
	0.5±5.98mmHg
		MSE = 51.8
		R = 0.9796

CP, DT	1.12±6.9144 and
	0.4556±7.155 mmHg
		MSE = 52
		R = 0.98

CP	0.3465±7.74 mmHg
	0.66±6.35 mmHg
		MSE=50.5
		R = 0.98

DT	6.07+7.2 mmHg
	3.02+8.3 mmHg
		MSE = 84
		R = 0.9749

SUT	79.9+19 mmHg
	31+3 mmHg
		MSE = 3872
		R = 0.788

8. Conclusion

As shown in this report there is enough information present in the PPG signal to identify the BP of a person. The regression plots look encouraging enough to show that one can use the PPG signal to estimate the Blood Pressure. In addition once three parameters are used we are within the limits required for good Blood Pressure measurement, so more parameters are not really required.

Neural Network outperforms linear incremental gradient descent by enough margin that this should be method considered for future studies.

However the fundamental problem now lies with the pre-processing step which in this work so far requires some manual input. In addition many waveforms were deemed too difficult for the algorithm to handle and hence discarded manually. One example of waveform’s which do not work is given in the Appendix. Future work would be to find good pre-processing algorithms to overcome this issue. Currently the reason the results look good is because of having to discard waveforms which could not be handled by the pre-processing step. Future work needs to be able to handle more waveforms.

9. Appendix

Some interesting waveforms that were observed are captured here to help future work. Currently the pre-processing step cannot handle the presence of a double peak. A few ABP signals showed the waveform given in the figure below. Reference [9] explains the reason behind this waveform shape

"In *aortic regurgitation,* the arterial pressure wave rises rapidly, pulse pressure increases, and diastolic pressure is low, owing to the runoff of blood into the left ventricle as well as the periphery during diastole. Because of the large stroke volume ejected from the left ventricle in this condition, the arterial pressure pulse may have two systolic peaks *(bisferiens pulse)* (see **Fig. 30–22)**. These two peaks represent separate percussion and tidal waves, with the former resulting from left ventricular ejection and the latter arising from the periphery as a reflected wave"

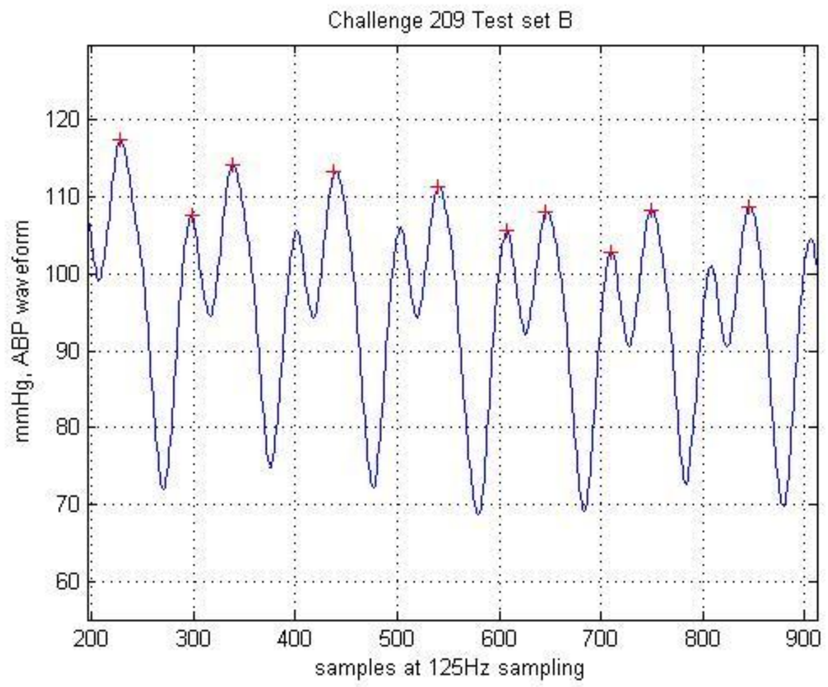


Figure 10 Example of an input ABP waveform that was not used for this project

References

[1] X. F. Teng and Y. T. Zhang, "Continuous and noninvasive estimation of arterial blood pressure using a photoplethysmographic approach," Proc. of 25th Annual Inter. Conf. of the IEEE Engineering in Medicine and Biology Society, Cancun,Mexico, 2003, pp. 3153–3156

[2] Y. Yoon, G. Yoon, "Non-constrained blood pressure measurement by photoplethysmography," Journal of the Optical Society of Korea, vol.10, no.2, pp.91-95, June 2006.

[3] Yu. Kurylyak, F. Lamonaca, D. Grimaldi, "A neural network-based method forcontinuous blood pressure estimation from a PPG signal," in Proc. IEEE International Instrumentation and Measurement Technology Conf., Minneapolis(MN), 2013

[4] Yu. Kurylyak et al., "Photoplethysmogram-based Blood Pressure Evaluation using Kalman Filtering and Neural Networks", in Proc. IEEE International Symposium onMedical Measurements and Applications, 2013.

[5] Rohan Samria et al., "Noninvasive Cuffless Estimation of Blood Pressure using Photoplethysmography without Electrocardiograph Measurement",in IEEE TENSYMP 2014.

[6]Satya Narayan Shukla, M.TECH Project mid-term report. Project title "Noninvasive Cuffless Blood Pressure Estimation from PPG signal"

[7] D. Deriso, N. Banerjee, A. Fallou, "Extracting vital signs from video". 2013 project for CS229

[8] Mohamed Elgendi, "On the Analysis of Fingertip Photoplethsymogram Signals", Current Cardiology Review

[9] http://web.squ.edu.om/med-Lib/MED_CD/E_CDs/anesthesia/site/content/v03/030267r00.HTM

[10] Training Feedforward Networks with the Marquardt Algorithm – Martin Hagan and Mohammad Menhaj, IEEE transactions on Neural Networks, 1994

[11] Sparse Autoencoder – CS294A Lecture Notes, Andrew Ng

[12] http://www.eng.auburn.edu/~wilambm/pap/2011/K10149_C012.pdf Levenberg–Marquardt

Training Notes