CHAPTER 6

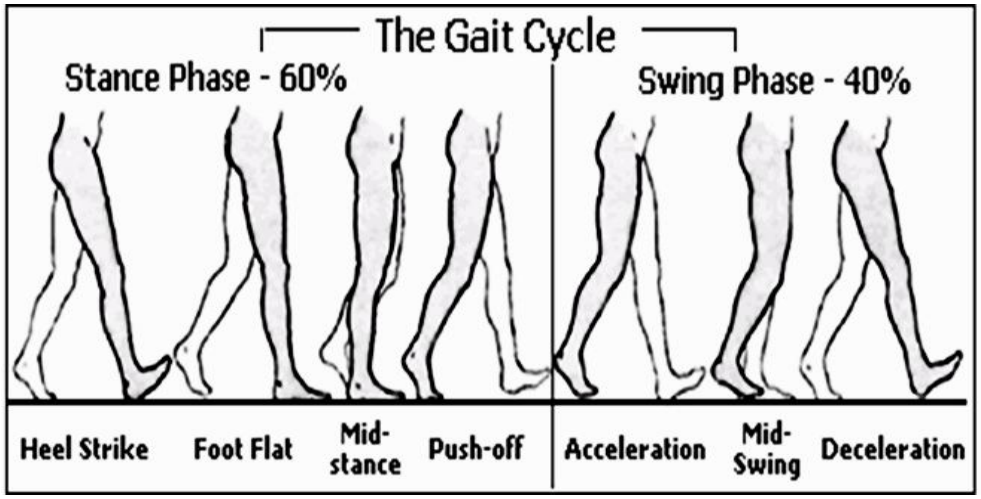
Q-BACKPROPAGATED TIME DELAY NEURAL NETWORK CLASSIFIER FOR DIAGONSING PARKINSON'S DISEASE

Parkinson's Disease (PD) is a movement disorder that affects the patient's nervous system and healthcare applications mostly uses wearable sensors to collect these data. Since these sensors generate time stamped data, analyzing gait disturbances in PD becomes a challenging task. The objective of this contribution is to develop an effective Clinical Decision Making System (CDMS) that aids the physician in diagnosing the severity of gait disturbances in PD affected patients. This chapter presents a Q-Backpropagated Time Delay Neural Network (Q-BTDNN) classifier that builds a temporal classification model. This classification model performs the task of classification in CDMS. The proposed Q-learning induced Backpropagation (Q-BP) training algorithm trains the Q-BTDNN by generating a reinforced error signal. The network's weights are adjusted by backpropagating the generated error signal.

6.1 PARKINSON'S DISEASE TRAITS

Parkinson's Disease (PD) affects the nervous system of the patients resulting in the postural instability or walking disturbances (Davie 2008). The symptoms of PD include muscle stiffness, tremors, and changes in speech and gait. Many research studies provide detailed investigations about the PD symptoms (Ahlrichs & Lawo 2013). In general, a gait cycle also referred as

stride contains one stance phase and one swing phase (Root2Being, Davie 2008). The stance phase represents the period at which the foot strikes on the ground. The swing phase represents the period at which the same foot lifts up the floor. Figure 6.1 illustrates the gait cycle (Root2Being, Davie 2008). A normal person's walking constitutes repetition of this gait cycle and a normal human being approximately takes 60% of stance phase and 40% of swing phase. PD patients often show disturbances and variations in this gait cycle. This work analyses the severity of gait disturbances in PD.



Source (Root2Being)

Figure 6.1 Gait Cycle

Frenkel-Toledo et al. (2005a); Yogev et al. (2005); Frenkel-Toledo et al. (2005b); Hausdorff et al. (2007) have presented an experimental study that examines the associations between the walking speed and variations in the gait. The authors have observed that for PD patients, there is a decrease in the stride length and average swing time. Similarly, it has been observed that the stride and swing time variations increases for PD patients. The impact of PD in terms of movement disabilities are measured using several rating scales namely Unified Parkinson Disease Rating Scale (UPDRS), modified UPDRS and Hoehn & Yahr Scale (Fahn & Elton 1987; Hoehn & Yahr 1967).

Giuberti et al. (2015) have evaluated the severity level of PD by characterizing the leg swiftness task. The authors have investigated an association between the angular amplitude and speed of thigh motion with UPDRS scores.

Though, there are several studies done to analyze the movement disorders in PD patients there are still many challenging areas of research in this domain due to the time stamped nature of PD data acquired through most of the wearable sensors.

6.2 COMPUTER ASSISTED DECISION SUPPORT SYSTEM FOR DIAGONSING PARKNISON'S DISEASE: BACKGROUND

Cole et al. (2014) have presented dynamic machine-learning algorithms for monitoring the tremor severity and dyskinesia by analysing signals collected from PD patients wearing small numbers of hybrid sensors. The authors have designed dynamic algorithms in pattern classification, neural networks, support vector machines and Hidden Markov Model (HMM). The experimental data were collected from eight PD patients and four healthy individuals through wearable sensors by allowing them to do unplanned and unrestricted daily living activities. The experimental results show that the performances of all the presented dynamic algorithms are equally effective in reducing the error rates.

Wang et al. (2012) have presented an incremental learning classification model based on fuzzy clustering algorithm and probabilistic neural networks for sensor-based human activity recognition. The presented classifier has the ability to adapt and incrementally learn from the training data. The system effectively performs classification with incremental data and proves its efficiency in terms of its learning ability and accuracy. Mazilu et al. (2013) have investigated the performance of feature learning process for detecting Freezing of Gait (FOG) in PD patients. The authors have provided a

comparison on feature learning approaches based on time-domain and statistical features to unsupervised features. The comparison was done based on principal components analysis. The experiments were conducted with acceleration data acquired from ankle of patients suffering with FOG. The experimental results prove that the statistical and time-domain features outperform the unsupervised feature extraction process.

Michael et al. (2014) have presented an evolutionary algorithm, namely Sliding Window Genetic Programming (SGP) and Artificial Biochemical Networks (ABN) to classify the movement characteristics of Parkinson disease patients. SGP is used to capture movement patterns within a cycle. ABN is used to capture dynamical patterns occurring during time scales. Two-thirds of the clinical recordings are placed in a training set for fitness evaluation. The other third of the data is used to evaluate the classifier. Though, both the techniques effectively classify PD patients and control patients the SGP classifier outperforms ABN. The evolutionary algorithms are recommended since it produces patterns that human might not notice. Ene (2008) has presented an application of Probabilistic Neural Network (PNN) for classifying healthy and PD subjects. The PNN model based on three searches namely incremental search, Monte Carlo search and hybrid search were used in the classification process. The experimental study was conducted with biomedical voice measures data for 25 PD patients and 6 normal persons obtained from University of California, Irvine(UCI) repository. From the experimental results, it can be inferred that there is no significant differences between three searches, however; the use of hybrid heuristic approach has improved the classification results.

Little et al. (2008) have presented a classification technique named Kernel based Support Vector machine to diagnose the PD by identifying dysphonia. For experimentation, the authors used sustained phonations from 23 PD patients and 8control persons. It was observed that the new dysphonia measure introduced such as pitch period frequency along with another ten

measures provides improved classification accuracy, which is recommended in many telemonitoring applications. Rigas et al. (2012) have presented a study to illustrate that a Hidden Markov Model (HMM) was well suited for identifying tremors since they mostly represents temporal dependencies. They have experimented with ten patients and thirteen control subjects daily activity accelerometer data. Djuric-Jovicic et al. (2010) have presented a thresholding technique and a neural network to classify PD patients based on their walking patterns. This distinguishes the normal walk and shuffling steps. For experimentation, the data was acquired using a set of six inertial measurement units attached to the subjects' legs (i.e. thigh and shin) as well as their feet. The movements of four patients for thirty minutes were collected and used to train a neural network. The error rate of the training process obtained depends on the choice of threshold.

Das (2010) has presented a comparative study about various classification methods, namely Decision Tree, Neural Networks, DMneural and Regression for diagnosing PD disease. For experimentation, the authors have used biomedical voice measurements from PD patients who are suffering from speech disorder. From the experimental results, it was observed that neural network outperforms other classifiers in terms of its classification accuracy. Ahlrichs et al. (2013) have presented a detailed review that describes various techniques used in diagnosing PD based on motor symptoms from times series data. The authors have provided detail descriptions about the accuracies and error rates with respect to the experimental data they have considered. Waibel (1989) has proposed a time delay neural network for identifying the temporal relationships among the acoustic-phonetic features.

Comparing to the works discussed the proposed work is different in following ways: This contribution presents a reinforced Q-Learning Backpropagation (Q-BP) algorithm to train the TDNN in an incremental way. The difference between the traditional BP and Q-BP lies in the mode of

propagating the error and adjusting the weights. In Q-BP instead of directly backpropagating the error, a reinforced error signal is generated using a Q-learning principle and then the error is backpropagated. The weight updations are done based on this reinforced error signal. The generated reinforced backpropagated error signal adjusts the network weights. Time-stamped gait patterns observed for each subject are taken into consideration for diagnosing the severity conditions of gait disturbances in PD.

6.3 METHODOLOGICAL FRAMEWORK FOR Q-BTDNN

The PD dataset used in this work is a time series data that describes a temporal sequence of observations on each subject walking pattern. A detailed description of the dataset was provided in Chapter 4. To build a classification model, this contribution presents a Q-BTDNN classifier that is trained using the proposed Q-BP algorithm. The trained classification model is used for predicting the gait disturbances in PD. The following sections provide a detailed description about the Q-BTDNN structure, the proposed Q-BP algorithm and its application in predicting the severity of gait disturbances in PD.

6.3.1 Q-BTDNN Structure and Q-BP Learning Algorithm

The traditional TDNN (Waibel 1989) is a biologically inspired neural network that is used to model time series data. TDNN interprets the temporal sequence effectively by relating the inputs in different time points. Q-BTDNN presented in this work is a feed forward time-delay neural network in which the training is done using the proposed Q-learning induced back-propagation (Q-BP) technique. Q-BP functions in an incremental way by combining the relative advantages of both the reinforced Q-learning (Jang 1996) and Back-propagation (William et al. 1986). The network structure for Q-BTDNN shown in Figure 6.2 includes three layers, namely tapped delay input layer, computational layer and reinforced feedback layer.

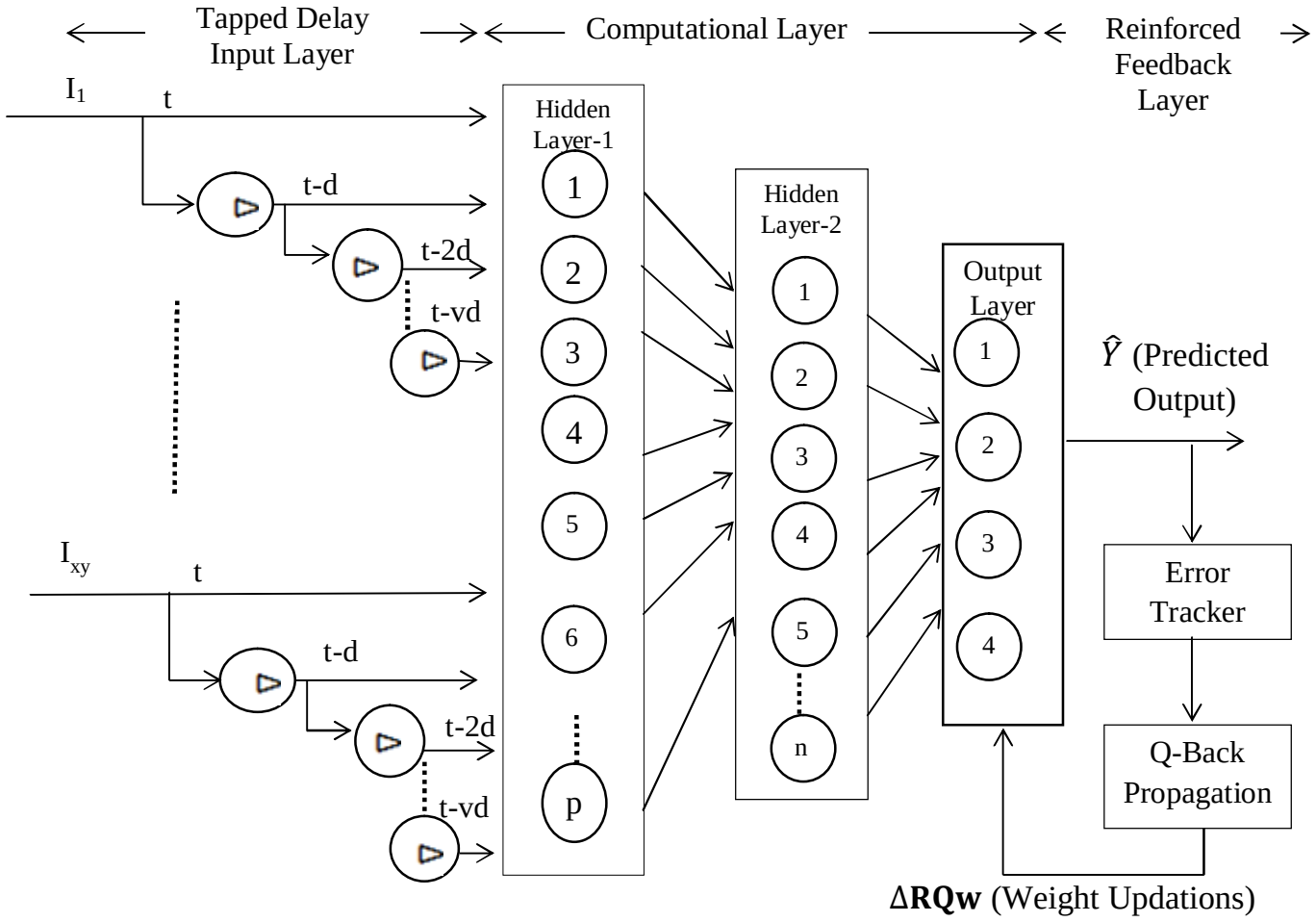


Figure 6.2 Q-BTDNN Structure

The tapped delay input layer feeds the inputs from a time series data. The input ($I_{1...}$,I_{xy}) refers to a clinical examination (or an attribute) that describes the medical test taken from a person (subject) where I_1 refers to first input attribute, I_{xy} refers to the xy^{th} input attribute. Each input is observed successively at different time points (t, t-d, t-2d..t-ad), where 't' refers to the time point, 'd' refers to the delay between each time point.

The computational layer comprises of hidden and output layers. The configuration of this layer relies on the chosen application. The error tracker keeps track of the weights for which minimum error is achieved. The reinforced feedback layer generates a reinforced error signal using the Q-learning technique, which is back-propagated on the network to adjust the network's weights. The traditional Backpropagation (BP) learning technique (William et al. 1986) minimizes the classification error by adjusting the network's weights through back-propagating the error obtained for each training input instance. A Q-BP learning algorithm trains the Q-BTDNN. Q-BP algorithm includes three major functionalities namely forward input propagation with delay, reinforced error back propagation and weight updation.

The forward input propagation with delay includes the process involved in forwarding the temporal inputs to Q-BTDNN in the forward direction with a delay. The reinforced error backpropagation process generates a reinforced error signal. The network weights are adjusted based on the generated error signal. The working procedure of Q-BP algorithm for training the Q-BTDNN is detailed below. The step 1 initializes the network weights randomly. The step 2 reads the input sequences. The step 3 to 4 summarizes the forward input propagation with delay. The step 5 to 8 summarizes the process involved in computing the reinforced error back propagation and weight updation.

// Initialization

Step 1: Initialize the weights of the network randomly with values ranging from -1 to 1.

Step2: Read the temporal sequence of each subject in TDNN from 16 sensors with a delay of 0.01 seconds.

// Forward input propagation with delay

Step3: For every node in the input layer its output equals the inputs along with the time delay lines given to it.

Step4: For every node (j) in the hidden layer, compute its output ($O_j(t)$) using the Equation (6.1).

$$O_j(t) = \varphi(\sum_{i=1}^{m} \sum_{g=0}^{l} RQw_{ij} x_i(t-g)) + \theta_j \quad (6.1)$$

where RQw_{ij} is the network weight adjusted using Q-learning, $x_i(t-g)$ inputs at the time $(t-g)$ for i[th] node in the input layer, m refers to the number of nodes in the input layer, g refers to the delay lag for each input sensor observation. θ_j is the bias and φ denotes the activation. For every node (k) in the output layer, compute its output ($O_k(t)$) using the Equation (6.2).

$$O_k(t) = \varphi(\sum_{j=1}^{h} RQw_{ik} O_j(t)) + \theta_k \quad (6.2)$$

where RQw_{ik} is the networks weight adjusted using Q-learning, $O_j(t)$ output from the j[th] hidden node, h refers to the number of hidden nodes, θ_k is the bias and φ denotes the activation.

// Reinforced Error Backpropagation (Q-Learning induced Error propagation)

Step5: For each node in the output layer compute its error based on the three functions, namely Q-function, reinforcement function and weight using the step 6, 7 and 8 using Q-learning method.

Step6: Q-function is computed when every subject's observation is given to the network. The computed Q- value defined in the equation (6.3) represents the utility function for the current patient state and the result of the action performed.

$$QW_{t+1}(S_t, a_t) = QW_t(S_t, a_t) + \lambda_t(S_t, a_t).(Ri_{t+1} + \gamma \max_a QW_t(S_t, a_t) - QW_t(S_t, a_t)) \quad (6.3)$$

where $QW_t(S_t, a_t)$ is the old Q-value, S_t and a_t refers to state and action, $\alpha_t(S_t, a_t)$ is the learning rate, Ri_{t+1} is the reward value, γ is the discount factor, $\max_a QW_t(S_t, a_t)$is the estimate of optimal value. In this work we refer three actions of adjusting weights namely Q-dependent, target value dependent, gradient descent, gradient descent with momentum. The estimate of optimal weights is identified by finding best minimal errors returned by taking any of these actions.

Step 7: The reinforcement signal (Rs_i), which represents the reward or penalty, is generated from the outcome of the error and is defined in the equation (6.4). A parameter named min_errortracker (MinE) is maintained to keep track of the weights for which minimum error is achieved. Initially the min_errortracker is assigned with first error generated with initialized random weights. As the network training process starts, it reassigns its value by tracking the latest local best optimal weight.

$$Rs_i = \begin{cases} 1, & \text{CurrE} < MinE \\ -1, & \text{CurrE} \geq MinE \end{cases} \quad (6.4)$$

The reinforcement signal assigns a reward (1) when it finds that the current predicted Error (CurrE) is less than MinE otherwise it sends a penalty signal (-1).

//Weight Updation

Step 8: Weight updations is done based on the generated reinforcement signal. If reward signal is activated then the network weights are adjusted by adding Q-value to the all current weights as defined in the Equation (6.5).

$$RQw_{ij} = RQw_{ij} + QW_{t+1}(S_t, a_t) \quad (6.5)$$

If the penalty is activated then the combination of Q-value and propagated error is averaged and used in the weight updation as defined in the Equation (6.6).

$$RQw_{ij} = RQw_{ij} + (QW_{t+1}(S_t, a_t) + \text{CurrE}) \quad (6.6)$$

Step9: The network training stops when the networks weights provides minimum error rate.

6.3.2 Application of Q-BTDNN to Diagnose Gait Disturbances in Parkinson's Disease

Q-BTDNN classifier was applied to diagnose the severity of gait disturbances in PD. The gait data used for experimentation is acquired from three PD studies (Yogev et al. 2005; Frenkel-Toledo et al. 2005b; Hausdorff et al. 2007). The description of the data and its method of data acquisition are discussed in the Chapter 4. The proposed work uses a network structure that includes one tapped delay input layer and computational layer comprising of two hidden layers and one output layer. The tapped delay input layer feeds the

successive observation of each 16 input sensors (Lg_1, Lg_2, Lg_3, Lg_4, Lg_5, Lg_6, Lg_7, Lg_8, Rg_1, Rg_2, Rg_3, Rg_4, Rg_5, Rg_6, Rg_7, Rg_8) recorded from the left leg and right leg of the subjects at different time points. During the network training process, this works considers 12000 time stamped observations from 16 sensors for each subject.

The computational layer comprises of two hidden layers with 240 hidden nodes and 120 hidden nodes respectively, and an output layer with four nodes. The reinforced feedback layer generates a reinforced error signal using the Q-learning technique, which is back-propagated on the network to adjust the network's weights. The input layer of Q-BTDNN forms a 16 x 12000 vector. The first hidden layer consists of 240 nodes arranged in 2 X 120 vector. The second hidden layer consists of 120 nodes arranged in 2 x 60 vector. The output layer consists of four nodes arranged in 1 X 4 vector. Each 100 frames in the input layer are mapped to one frame in the first hidden layer. Every 2 frames in the first hidden layer are mapped to a node in the second hidden layer. Each 15 frames in the second hidden layer were connected to one frame in the output layer. For each subject instance, the network computes an output.

In forward input propagation with delay, the data from 16 sensors observed from the left and right legs of the subjects are fed forward to the time delay network with the delay of 0.01 seconds. These inputs are forwarded to the Q-BTDNN using steps 1 to 4. A reinforcement based Q-learning approach is used to generate and back-propagate the error signal using steps 5 to 8.

6.4 EXPERIMENTAL SETTINGS

This section discusses the experimental settings and evaluation criteria used to assess the effectiveness of the Q-BTDNN classifier with three research study dataset. Experiments were conducted with 93 PD subjects and 73 normal subjects; each observed data record corresponds to the walking

patterns observed in 2 minutes' walk using wearable sensors. Figure 6.3 shows the graphical plot that illustrates the total force recorded for 15 seconds from the 16 sensors. Figure 6.3 (a) and (b) describes the total force recorded in the left and right leg of a PD affected subject. Figure 6.3 (c) and (d) describes the total force recorded in the left and right leg of a normal subject. Stride refers to the period the foot strikes the ground goes to swing and the foot in the same leg strikes the ground. From the Figure 6.3 (a), (b), (c) and (d) it can be inferred that when a person is in stance state the force recorded will be higher compared to the moment he or she is going to swing state. PD affected patients show a high increase in the average number of strides within an observed duration compared to the normal control patients. The variations in stance, swing and stride time of the PD patients show a high impact in identifying the severity of the disease.

The input data from 16 sensors are propagated in the Q-BTDNN with a delay of 0.01 sec. The records were sampled at the rate of 100 samples per second that is observed for two minutes. The implementations and experimentations were carried out using the neural network tool box in MATLAB 2013 (Demuth & Beale 2013). A 10-fold cross validation was used for experimental evaluation. A 10-Cross-validation model assessment technique randomly split the dataset into 10 subsets of approximately equal size. For training, 9 subsets were used for training and remaining one subset is used for testing. The entire process is repeated for 10 times. To make this evaluation unbiased it is required to make several runs of 10-fold cross validation. Therefore, in this work the classification model was assessed using 10-fold cross validation repeated for 5 independent runs.

To select the best fit parameter values for Q-BTDNN training, a series of experiments were conducted with different combinations of network parameters based on hidden layers. The parameter for which the network in its training process gives a minimal RMSE were selected. Based on the outcome of experiments, the computational layer of Q-BTDNN is configured

with two hidden layer and an output layer for PD gait database. The first hidden layer has 240 hidden nodes arranged in 2 X 120, the second hidden layer has 120 nodes arranged in 2 X 60 and an output layer with four nodes. The four output nodes represent the class label minimal, mild, moderate and severe. These class labels in the training records describe the severity of gait disturbances in PD based on the Hoehn and Yahr Scale (Hoehn & Yahr 1967). The Hoehn and Yahr Scale represent the PD stages in the scale of 1.5 to 2.5 based on the motor symptoms.

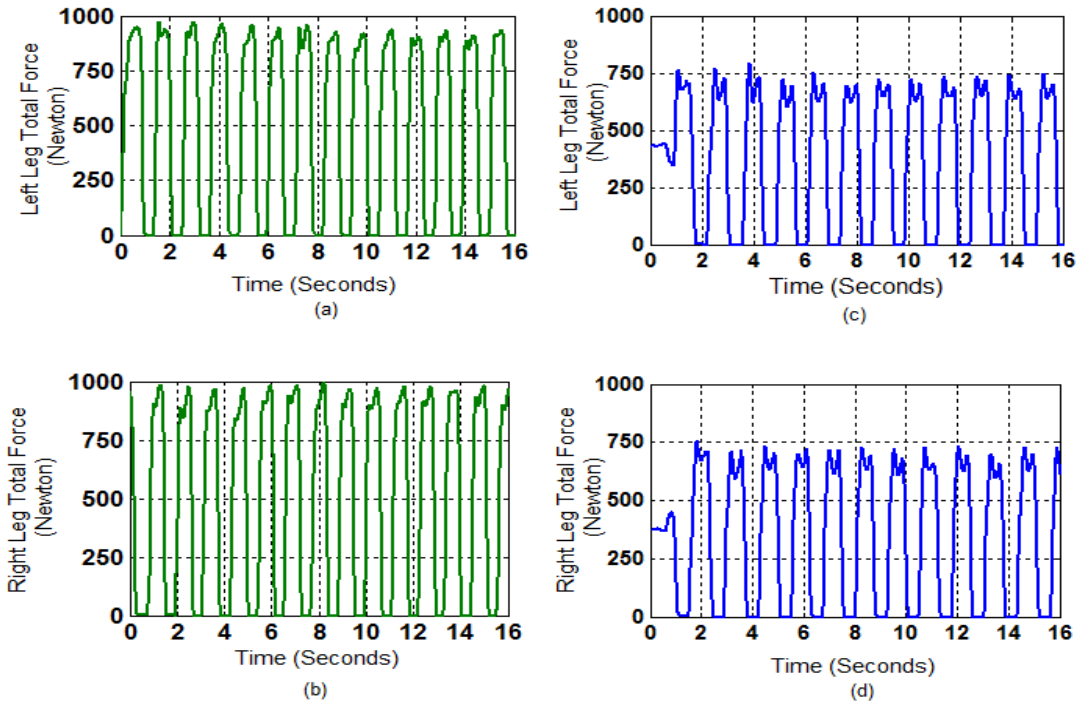


Figure 6.3 Total Force (a) Left Leg of PD Patient, (b) Right Leg of PD Patient (c) Left Leg of Normal Person (d) Right Leg of Normal Person

Accordingly, this works considers the class label minimal for scale rating 1 to 1.5, mild for 1.5 to 2, moderate for 2 to 2.5 and severe for more than 2.5. A learning rate of 0.01 and sigmoid activation function was chosen for Q-BTDNN training. The network output scales the value to be in the range of 0 to 1. The output in four nodes is interpreted as 1000 for minimal, 0100 for mild, 0010 for moderate and 0001 for severe. The realizations of the presented classifier with different combinations of hidden units are discussed

below. The work was tested with network configuration of one hidden layer with 240, 120 and 80 hidden nodes. For one hidden layer with 240 nodes, a vector space of 2 x 120 was considered. Each 100 frames in the input layer were mapped to one frame in the hidden layer. Each 30 frame in hidden layer is mapped to one node in the output layer. For one hidden layer with 120 nodes a vector space of 2 x 60 was considered. Each 200 frames in the input layer were mapped to one frame in the hidden layer. Each 15 frame in hidden layer is mapped to one node in the output layer. For one hidden layer with 80 nodes a vector space of 2 x 40 was considered. Each 300 frames in the input layer were mapped to one frame in the hidden layer. Each 10 frame in hidden layer is mapped to one node in the output layer.

For two hidden layers of 240 and 120 nodes, the first hidden layer forms 2 X 120 vector and Second hidden layer forms 2 X 60 vector. Each 100 frames in input layer were mapped to one frame in the first hidden layer, each 2 frames in the first hidden layer is mapped to one frame in second hidden layer, each 15 frame in the second hidden layer is mapped to one node in output layer. For two hidden layers of 120 and 60 nodes, the first hidden layer forms 2 X 60 vector and second hidden layer forms 2 X 30 vector. Each 200 frames in input layer is mapped to one frame in the first hidden layer, each 2 frames in first hidden layer is mapped to one frame in second hidden layer, each 13 frame in second hidden layer is mapped to one node in output layer. For two hidden layers of 80 and 40 nodes, the first hidden layer forms 2 X 40 vector and second hidden layer forms 2 X 20 vector. Each 300 frames in input layer is mapped to one frame in first hidden layer, each 4 frames in first hidden layer is mapped to one frame in second hidden layer, each 5 frame in second hidden layer is mapped to one node in output layer.

6.5 RESULTS AND DISCUSSIONS

This section discusses the results and findings obtained from the proposed Q-BTDNN classifier with the three PD study datasets (Yogev et al.

2005; Frenkel-Toledo et al. 2005b; Hausdorff et al. 2007). The best fit values for the network parameters in training process are selected based on the realization of huge number of network architectures and experimentations. Table 6.1 shows a few results of this realization during the network construction and parameter selection process. The root mean square error (RMSE) obtained at 100, 150 and 200 epochs was shown for the different combinations of the number of nodes in the computational layer of the network structure. The network stabilizes itself and RMSE value reaches minimum at 200 epochs for the Parkinson's dataset and hence the training was stopped.

Table 6.1 RMSE values vs Epochs Observed for Varied Combinations in Q-BTDNN Structure

Parkinson's Disease Dataset	Hidden Layers	Hidden Layer 1 nodes	Hidden Layer 2 nodes	RMSE (x 10^{-1})		
				100 Epochs	150 Epochs	200 Epochs
Yogev et al. (2005)	1	240	--	0.765	0.713	0.691
		120	--	0.791	0.726	0.611
		80	--	0.801	0.782	0.776
	2	240	120	0.414	0.358	0.279
		120	60	0.647	0.616	0.597
		80	40	0.698	0.679	0.623
Frenkel-Toledo et al. (2005b)	1	240	--	0.703	0.696	0.615
		120	--	0.765	0.723	0.698
		80	--	0.799	0.781	0.786
	2	240	120	0.382	0.328	0.239
		120	60	0.567	0.534	0.512
		80	40	0.618	0.597	0.555
Hausdorff et al. (2005)	1	240	--	0.798	0.711	0.701
		120	--	0.801	0.785	0.713
		80	--	0.823	0.811	0.791
	2	240	120	0.433	0.378	0.299
		120	60	0.667	0.613	0.612
		80	40	0.638	0.653	0.703

A 10-fold-cross-validation repeated for 5 runs generates test and training set for evaluating the classification model. The results obtained from each fold is averaged and considered for comparative analysis. Figure 6.4 depicts the graphical plot for illustrating the RMSE value obtained for various iterations during Q-BTDNN training for the three datasets (Yogev et al. 2005; Frenkel-Toledo et al. 2005 b; Hausdorff et al. 2007). It can be observed that at 200 epochs the networks RMSE value stabilizes.

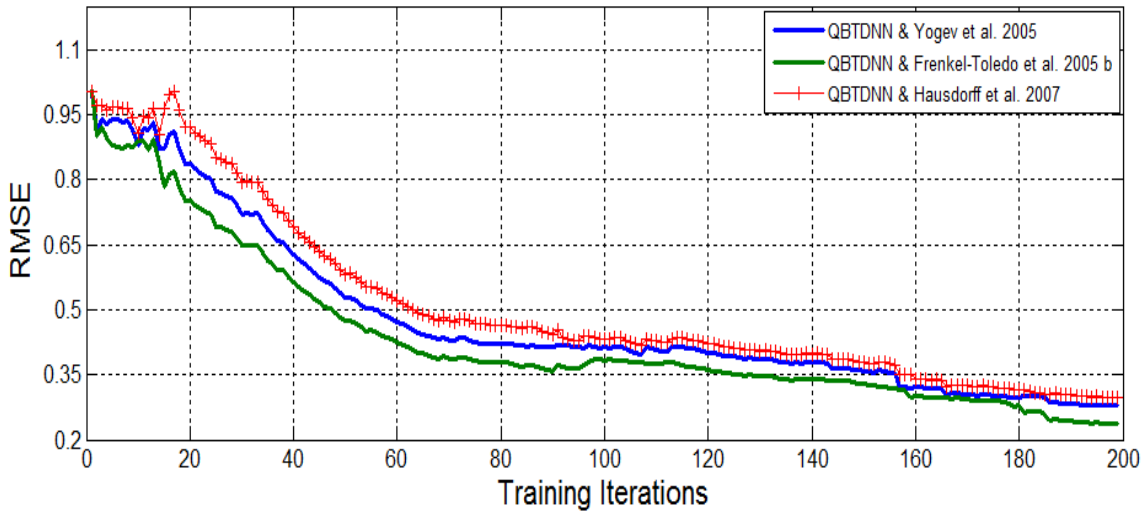


Figure 6.4 Q-BTDNN- Training Iterations Vs Root Mean Square Error (RMSE)

The classification results were evaluated with the following measures true positive rate (TPR), true negative rate (TNR), recognition rate (RR), misclassification rate (MR) and precision (Han & Kamber 2001). TPR refers to the proportion of subjects correctly classified as having PD. TNR refers to the proportion of subjects correctly classified, as not having PD. RR is the percentage of subjects correctly classified as PD or normal. MR is the percentage of subjects not correctly classified as PD or normal. Precision is the measure of percentage of records labelled as PD tuples are correctly as such. It is observed from Figure 6.4 that the RMSE value reaches minimum at 200 iterations and in this work the training terminates at this point. The

performance of the proposed Q-BP algorithm for TDNN is compared with other training algorithms, namely Levenberg-Marquardt (LM), Gradient Descent (GD), Gradient Descent with Momentum (GDM) and Scaled Conjugate Gradient (SCG) learning algorithms (Hagan & Menhaj 1999; Moller 1993; Hagan et al. 1996). Table 6.2 illustrates the classification results obtained from the proposed Q-BP and the other related learning algorithms.

Table 6.2 Comparison of classification results for parkinson's disease dataset

Parkinson's Disease Dataset	Network Learning Techniques	TPR	TNR	RR	MR	Precision
Yogev et al. (2005)	Q-BP *(Proposed)*	0.77	0.55	91.49	8.51	93.10
	Levenberg-Marquardt	0.69	0.48	80.85	19.15	82.76
	Gradient Descent	0.60	0.41	70.12	29.79	72.41
	Gradient Descent with Momentum	0.66	0.45	76.60	23.40	79.31
	Scaled Conjugate Gradient	0.63	0.41	72.34	27.66	75.86
Frenkel-Toledo et al. (2005b)	Q-BP *(Proposed)*	0.943	0.897	92.19	7.81	91.67
	Levenberg-Marquardt	0.886	0.793	84.38	15.63	83.78
	Gradient Descent	0.829	0.724	78.13	21.88	78.38
	Gradient Descent with Momentum	0.829	0.828	82.81	17.19	82.86
	Scaled Conjugate Gradient	0.800	0.724	76.56	23.44	80.00
Hausdorff et al. (2005)	Q-BP *(Proposed)*	0.74	0.83	90.91	9.09	89.66
	Levenberg-Marquardt	0.71	0.72	83.64	16.36	83.33
	Gradient Descent	0.60	0.66	72.73	27.27	65.63
	Gradient Descent with Momentum	0.66	0.76	81.82	18.18	79.31
	Scaled Conjugate Gradient	0.60	0.69	74.55	25.45	70.00

The obtained classification accuracy of Q-BP with TDNN for the data gathered in the study (Yogev et al. 2005) is 91.49% and for LM, GD, GDM and SCG are 80.85%, 70.12%, 76.6%, 72.34% respectively. For the data gathered in the study (Frenkel-Toledo et al. 2005b) the obtained classification accuracy is 92.19% and for LM, GD, GDM and SCG are 84.38%, 78.13%, 84.38%, 76.56% respectively. For the data gathered in the study (Hausdorff et al. 2007) the obtained classification accuracy is 90.91% and for LM, GD, GDM and SCG are 83.64%, 72.73%, 81.82%, 74.55% respectively. From the classification results, it has been observed that TPR, FPR, RR, MR and precision of Q-BP with TDNN outperforms other TDNN training algorithms like LM, GD, GDM and SCG.

In addition to the traditional classification metrics a statistical hypothesis testing was also carried out using paired t-test (Belciug & Gorunescu 2014; Zimmerman & Donald 1997). The test was carried out with significant level of 0.05 to identify whether there was any significant change in the error rates of Q-BP compared with Levenberg-Marquardt (LM), Gradient Descent (GD), Gradient Descent with Momentum (GDM) and Scaled Conjugate Gradient (SCG) learning algorithms. The ρ value of less than 0.05 is obtained, which proves that there is a significant change in the error-rates of Q-BP compared to Levenberg-Marquardt (LM), Gradient Descent (GD), Gradient Descent with Momentum (GDM) and Scaled Conjugate Gradient (SCG) learning algorithms. Thus, the evaluation results show that there is a significant improvement in the classification accuracy with the presented Q-BP learning algorithm and Q-BTDNN. The graphical illustration of the results of statistical assessment obtained for various significant level is shown in the Figure 6.5.

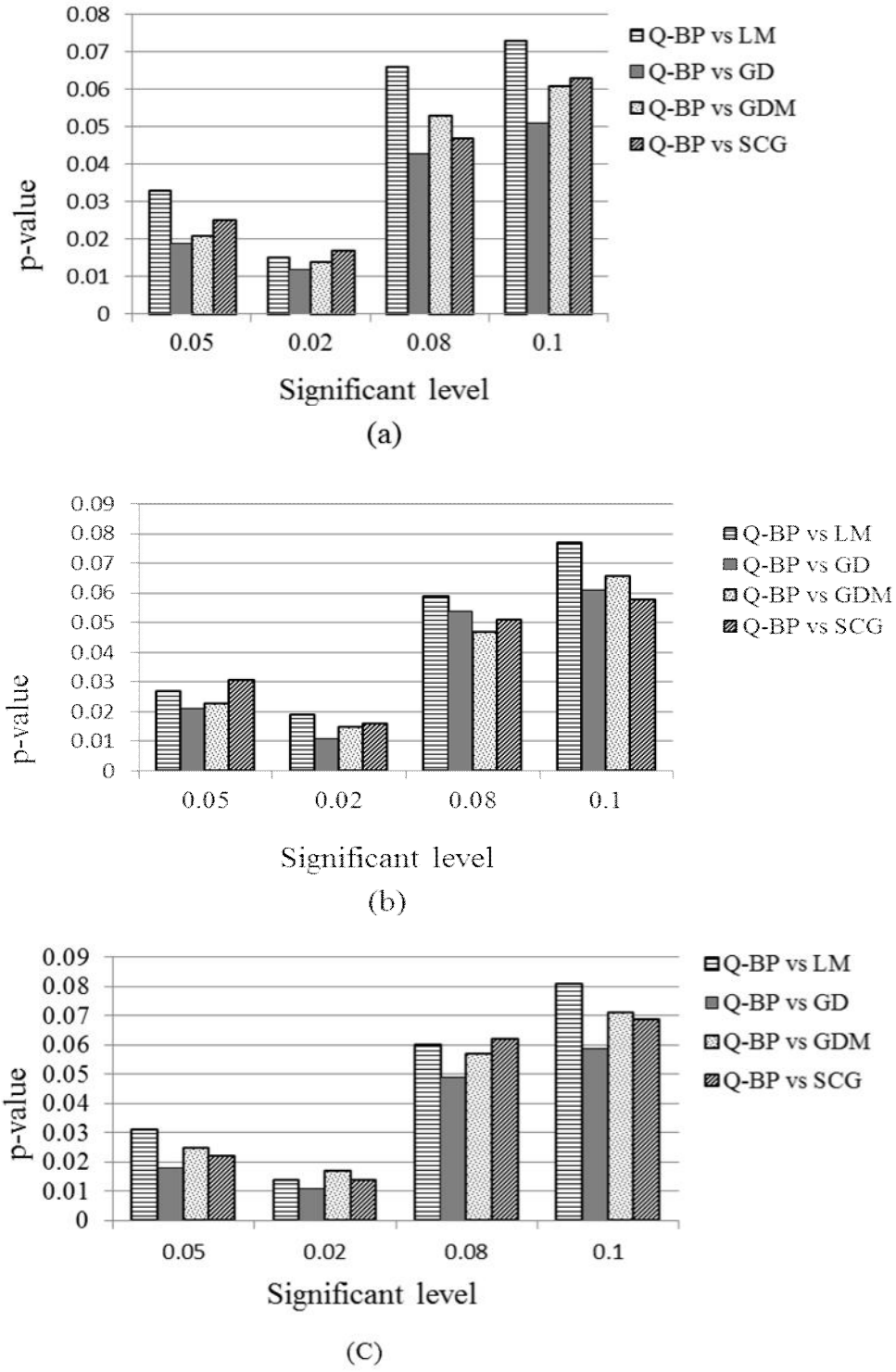


Figure 6.5 Results of Statistical Paired t-test on Classification Accuracies (a) Dataset: Yogev et al. 2005 (b) Dataset: Frenkel-Toledo et al. 2005b (c) Dataset: Hausdorff et al. 2007

The trained classification model is effective in diagnosing the gait disturbances in PD patients, which makes it recommended and applicable for practical usage. Thus, the trained classification model can be used in developing a clinical decision making system for assisting the clinician in diagnosing the severity of gait disturbances in PD affected patients.

6.6 CONCLUSION

This work presented a Q-BTDNN classifier that monitors and predicts the severity of gait disturbances in PD affected patients by analyzing the instabilities in the postures and walking patterns. A Q-Backpropagated (Q-BP) training algorithm has been proposed to train the Time Delay Neural Network for building a classification model that is used to develop a Clinical Decision Making System (CDMS). This CDMS is used to assist the physician in diagnosing the severity of gait disturbances in Parkinson's disease affected patients. The proposed work demonstrates its effectiveness on three PD study data (Yogev et al. 2005; Frenkel-Toledo et al. 2005b; Hausdorff et al. 2007). The experimental results shows, that the classification accuracy of Q-BTDNN for the three PD datasets is 91.49%, 92.19% and 90.91% accordingly. To prove the effectiveness and scalability, the system has been evaluated using large-scale clinical data which is not related to gait analysis. The experimental results showed improved classification accuracy and have proven the extendibility of the system with large-scale data. However, as a future work, the authors are investigating studies on extending the Q-BP algorithm to large-scale PD data analysis.