Master's Thesis

Automating the Fugl-Meyer Assessment with Supervised Machine Learning

Paul Otten

Department of Information and Communication Engineering

DGIST

2015

Master's Thesis

Automating the Fugl-Meyer Assessment with Supervised Machine Learning

Paul Otten

Department of Information and Communication Engineering

DGIST

2015

Automating the Fugl-Meyer Assessment with Supervised Machine Learning

Advisor: Professor Sang Hyuk Son

Co-Advisor: Professor Taejoon Park

Co-Advisor: Professor Jonghyun Kim

by

Paul Otten

Department of Information and Communication Engineering

DGIST

A thesis submitted to the faculty of DGIST in partial fulfillment of requirements for the degree of Master of Science in the Department of Information and Communication Engineering. The study was conducted in accordance with the Code of Research Ethics[1)].

11. 14. 2014

Approved by

Professor Sang Hyuk Son _____

Professor Taejoon Park _____

Professor Jonghyun Kim _____

1) Declaration of Ethical Conduct in Research: I, as a graduate student of DGIST, hereby declare that I have not committed any acts that may damage the credibility of my research. There include, but are not limited to: falsification, thesis written by someone else, distortion of research findings or plagiarism. I affirm that my thesis contains honest conclusions based on my own careful research under the guidance of my thesis advisor.

Automating the Fugl-Meyer Assessment with Supervised Machine Learning

Paul Otten

Accepted in partial fulfillment of the requirements for the degree of Master of Science.

12. 5. 2014

Head of committee Sang Hyuk Son (signature)

Prof. Sang Hyuk Son

Committee Member Taejoon Park (signature)

Prof. Taejoon Park

Committee Member Jonghyun Kim (signature)

Prof. Jonghyun Kim

MS/IC
201322020

Paul Otten, Automating the Fugl-Meyer Assessment with Supervised Machine Learning. Department of Information and Communication Engineering, 2015, 38p. Academic Advisors: Sang Hyuk Son and Taejoon Park, Co-Advisor: Jonghyun Kim

ABSTRACT

Evaluation of post-stroke hemiplegic patients is an important aspect of rehabilitation, especially for assessing improvement of a patient's condition from a treatment. It is also a necessary step to perform during clinical trials. The Fugl-Meyer Assessment (FMA) is one of the most widely recognized and utilized measures of body function impairment for post-stroke patients. We propose a method for automating the upper-limb portion of the FMA by gathering data from sensors monitoring the patient. Features are extracted from the data and processed by a machine learning algorithm. The output from the machine learning algorithm returns a value that can be used to score a patient's upper limb functionality. The machine learning algorithms tested in our system were Support Vector Machines (SVM) and Backpropagation Neural Networks (BNN). This system will enable automating and inexpensive stroke patient evaluation that can save up to 30 minutes per patient for a doctor, providing a time-saving service for doctors and stroke researchers.

Keywords: stroke assessment, Fugl-Meyer Assessment, machine learning, Support Vector Machine, Kinect

Contents

List of Tables

List of Figures

I. INTRODUCTION

The Fugl-Meyer Assessment (FMA) is a widely recognized method for performing a quantitative evaluation of a post-stroke patient's limb usage. It covers 5 domains: upper extremity motor, lower extremity motor, sensory, balance, and range of motion. The upper extremity motor section consists of 33 tests, which are movements performed by a patient (such as joint flexion or other limb movements). As the patient performs each movement, a score of 0 to 2 is given. The assigned scores generally follow these guidelines:

- 0: the patient cannot perform the movement at all
- 1: the patient can perform the movement partially
- 2: the patient can perform the movement faultlessly

The maximum score for the upper extremity motor section is 66. The test typically takes a clinician about 30 minutes to perform for each patient.

We propose a system design that uses sensor data to automatically score a subject according to the FMA specification without the presence of a doctor or clinician. The subject simply sits in a chair in front of the Kinect sensor and a display with the user interface. The user interface instructs the subject to perform a series of movements, playing a video guide for each one. As the subject performs each movement, sensors observe the movement and pass the data to the main application to be processed. Once the application extracts useful features from the sensor data, it classifies the movement with a machine learning algorithm, which gives a score of 0, 1, or 2 as its output. Our system supports 25 out of the 33 upper extremity motor tests (listed in Table 1), giving it a maximum possible score of 50. We believe that this provides a service to doctors treating stroke patients because it can save a significant amount of time for them. It also services researchers who are performing clinical trials on large numbers of stroke patients, since it gives an easy way for them to assess multiple stroke patients at periodic intervals with less effort than with currently available methods.

The main sensor that the system uses is the Microsoft Kinect, which has an infrared camera that can be used with the Kinect SDK [21] to gather 3D skeleton information of up to two people in its view. The skeleton information partially consists of 3D coordinates for 16 pre-defined joints at 30 frames per second. The Kinect has already been used for similar gesture recognition applications [17], [18] and can be used with additional sensors for our purposes.

In addition to the Microsoft Kinect, we used a pair of DG5-VHand Glove 3.0 sensors, originally developed for motion capture and gaming. This sensor is a glove that has flexion sensors sewn into the fingers. The glove also has an inertial measurement unit (IMU) with 9 degrees of freedom from its accelerometer, gyroscope, and magnetometer. This sensor is required because the Kinect skeleton information doesn't include any information about fingers or forearm pronation and supination. The VHand glove allows us to extend the number of upper extremity tests we can support. Our test coverage is further extended by the use of a simple pressure sensor. An overview of the sensors and system design are shown in Figure 1. The actual sensors we used in this design can be seen in Figure 2.

With these sensors, we record approximately 5 seconds of the subject's movement. After we gather the sensor data, we run it through feature extraction routines. Each test has its own feature extraction routine because different movements are scored based on different factors. Once we extract our features, we train a machine learning algorithm and use it to classify future movement recordings.

The two machine learning algorithms we tested were Support Vector Machines (SVM) and Backpropagation Neural Networks (BNN). SVM is a statistical machine learning approach that attempts to separate training data along a hyperplane. The hyperplane is created by training the SVM with training data. New data can then be classified based on its position relative to this hyperplane. BNN is another approach to machine learning, which tries to simulate a network of neurons to mimic the functionality of a brain. BNNs typically have 3 layers of nodes: an input layer, hidden layer, and output layer. When the input nodes are given a value, the nodes “fire” by returning values passed to the hidden nodes, which in turn “fire” by passing values to the output nodes. The output nodes represent the result of the classification. Each node has a weight and a bias value that influence the output it gives. These are the values that must be adjusted to train the neural network. The training algorithm, Backpropagation, is

based on the Gradient Descent algorithm, which is run iteratively to update the weights and biases. Once these values are set in such a way that the desired outputs are given within a specified error, the neural network can be used to classify future test data.

The main contribution of this thesis is to provide a system design that can be implemented in a hospital to save time for doctors when they need to evaluate the limb functionality of a patient. Our system currently covers 76% of the FMA upper extremity motor tests and it can be extended to support additional tests with additional sensors. Even with partial test coverage, our system saves a significant amount of time for doctors who must evaluate multiple patients (approximately 30 minutes per patient). We believe that the use of this system will also encourage more frequent assessment, allowing patients to track their progress weekly or monthly. This is because patients may use this system as a tool to assess themselves without having to use up a clinician's time.

Shoulder	Abduction
	Ext. Rotation
Elbow	Flexion
Forearm	Supination
Shoulder	Add/int. rotation
Elbow	Extension
Forearm	Pronation
Hand	Move to lumbar
Elbow 0^o	Pro/Supination
Elbow 90^o	Pro/Supination
Shoulder	Abduction $0\text{-}90^o$
	Flexion $0\text{-}90^o$
	Flexion $90\text{-}180^o$
Wrist	Elbow 90^o -wrist flex/extension
	Elbow 0^o -wrist flex/extension
	Circumduction
Hand	Finger mass flexion
	Finger mass extension
	Grasp a (hook)
	Grasp b (radial)
	Grasp c (opposition)
	Grasp d (cylinder)
	Grasp e (spherical)
Coordination/speed	Tremor
	Speed

Table 1: The tests supported by our system, listed in the order that they appear in [5].

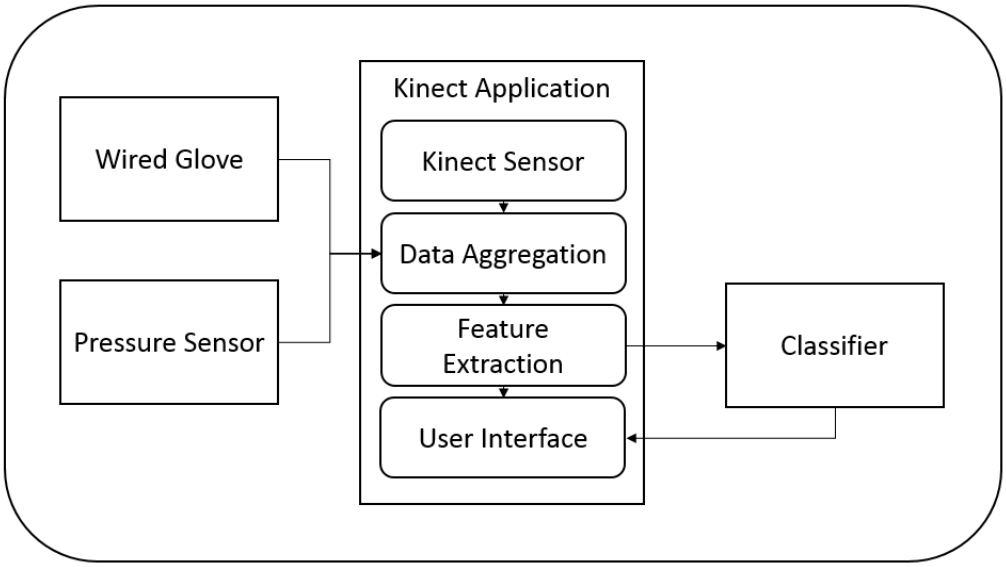
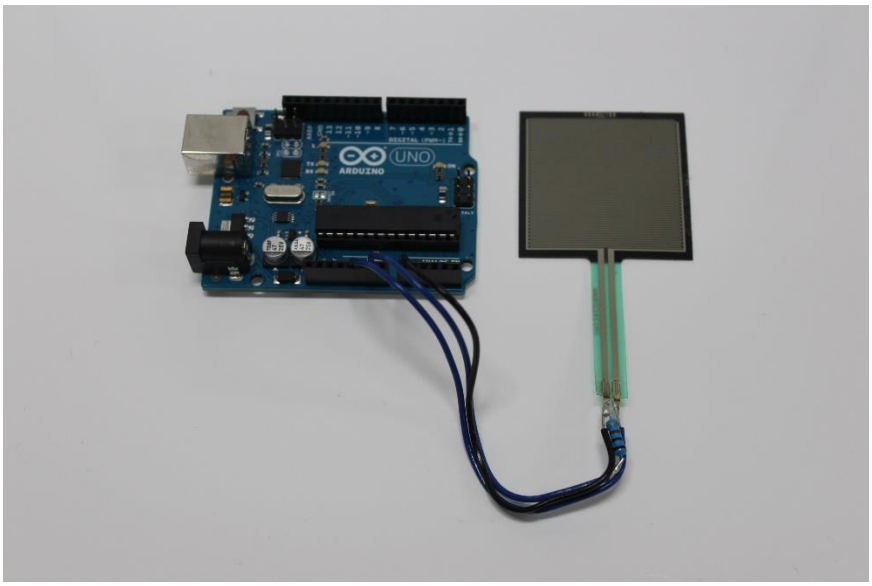


Figure 1: An overview of the system design showing the individual components.

(a)

(b)



(c)

Figure 2: The sensors that we used in our implementation of the proposed system. The main sensor is the Kinect (a). The wired data glove (b) and the pressure sensor (c) are used to extend the test coverage of the Kinect.

II. RELATED WORKS

The FMA was first proposed in [1] to provide the test movements and scoring methods. Papers [2], [3], and [4] review the assessment and its ability to reflect a meaningful quantitative score for a patient's limb functionality. These papers agree that stroke patient evaluation is a critical part of stroke rehabilitation and for stroke research.

One of the metrics used to evaluate the FMA is interrater reliability, which is a measure of how consistently clinicians give similar scores to a patient performing the movements. The authors in [3] found the FMA to have 98-99.5% interrater reliability. As another metric for reliability, [2] reports that intratester reliability is very high with total scores being consistent to within 0.001. These results were supported by [5], which found interrater reliability to be 93-99%. Papers [2], [3], and [4] all argue that the FMA is a reliable method of stroke patient assessment that represents statistically significant information.

According to [1], the FMA takes a doctor about 30 minutes to perform on a single patient. For many research experiments that involve a large number of patients, assessing each of these patients for experimental results can be a very time consuming and expensive process. Furthermore, [6] points out how stroke is the leading cause of death in China for people between the ages of 15 and 59. There are 2.5 million new stroke cases in China each year, with a total of 7.5 million current stroke survivors in need of rehabilitation and assessment. An automated version of the FMA could be used to decrease the time and expense by using multiple setups of cheap sensors to score multiple stroke patients.

The authors in [7] investigated the effect of home-based telerehabilitation tests on seven stroke patients, finding that the patients improved in the clinical and kinematic tests over time. The paper also concluded that their results suggested that there may be auxiliary benefits in cognitive function. In [8], the authors tested a telerehabilitation system based on the FMA and Wolf Motor Test (WMT). They found that the subjects' improvements were clinically and statistically significant, particularly with the FMA, WMT, and shoulder strength. The authors discussed that the elimination of commuting to the hospital gave patients more energy to perform their telerehabilitation, making it more effective.

With the large volume of patients that must be assessed with rehabilitation and clinical trials, there have been studies to determine the feasibility of automating the FMA in order to save time for the clinicians. In [9] and [10], the authors demonstrate that the Microsoft Kinect is very capable of recognizing 3D joint positions accurately enough to monitor a stroke patient and perform gesture recognition. The author in [9] compared the Kinect to another research-grade motion capture system called OptiTrack and found that the Kinect performed with similar levels of tracking accuracy. The authors in [11] developed a method of performing rehabilitation with the Kinect and a computer game. They came to the conclusion that this was a very effective and practical way to help stroke patients recover. In [12], the authors focused on the feasibility of automating the FMA using robotic arms, motion capture systems, and EMG sensors. They found that these devices could precisely capture movement data from stroke patients. However, their approach to automating the FMA was very expensive.

To further discuss the feasibility of automating the FMA, we can view the problem as a gesture recognition problem. Gesture recognition has been a well-researched problem over the past few years. Some of the latest studies use Kinect data with machine learning to automatically determine poses or actions of individuals. The authors in [17] used the Kinect with 4 different machine learning algorithms to classify between 8 different human poses with up to 100% accuracy. A similar paper used the Kinect with SVM to recognize between 8 different poses with similar accuracy [18]. Other papers, such as [19], [20] used the Kinect and machine learning to identify behaviors, such as aggressive behavior or dancing. These papers are evidence that it is highly feasible to use the Kinect and machine learning technologies for gesture recognition, which is the key to automating the FMA.

Given the feasibility of automated stroke patient evaluation, there have been many studies on actually implementing working systems to evaluate patients with sensor data. Papers such as [13] and [14] use sensors such as Inertial Measurement Units (IMU) and wired gloves to measure the movements of stroke patients during rehabilitation. They have shown that sensor data can be reliably read and be used in helpful ways to aid rehabilitation. The authors in [15] used a regression algorithm to automate some of the FMA tests with accelerometer data. Despite their success, their approach could only automate a very small number of FMA tests (4 out of 33 upper extremity motor tests). The system proposed in [16]

covered more tests (10 out of 33) and used the Kinect to perform motion capture. The results shown in this paper indicate that the Kinect is clearly a cheap and effective way to record and automatically score FMA test movements. However, the authors in [16] used principal component analysis for patient scoring, instead of a machine learning algorithm. They were also unable to create a system to fully automate the FMA, since they were only able to automate the scoring. However, this is definitely a step in the direction that we have gone with our paper, which can fully automate the process of collecting the data, pre-processing it, and scoring it with a machine learning algorithm.

III. SENSOR DATA

The FMA requires a patient to perform movements as tests in front of a clinician. The clinician then scores the test based on how well the patient performs the movement. Our approach to automating the assessment is to replace the clinician with a set of sensors and a user interface as shown in Figure 3. The sensors that gather data are a Kinect, a wired data glove, and a separate pressure sensor. Figure 4 shows how the system is set up in our laboratory.

Figure 3: The primary user interface for our system, which shows a graphical representation of the Kinect's sensor data

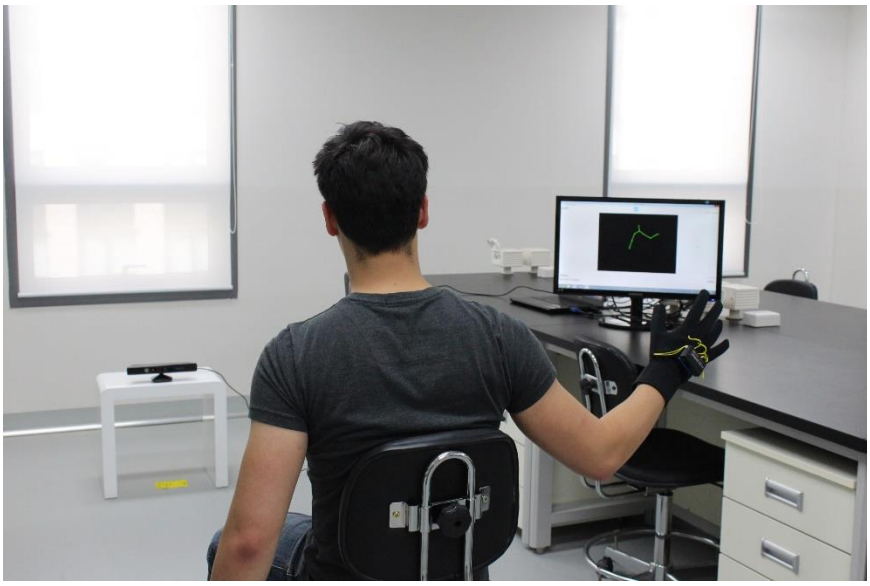


Figure 4: The lab setup where we recorded data from people performing the FMA.

3.1 Kinect

The primary sensor used for the evaluation is the Kinect, which gets a depth image of the person performing the test movements. The Kinect SDK allows us to use this data to get the 3D positions of a person's head, center chest, shoulders, elbows, and wrists. These 3D joint positions are recorded at 30 frames per second as the subject performs each of their test movements.

The Kinect is directly integrated into the main application, which contains a buffer for Kinect skeleton frames. Once the user initiates sensor recording with a voice command, the system starts storing skeleton frames. After storing every 10th frame, the buffer is processed to check if a movement has occurred. This is done by checking the distances that the wrists have traveled three times a second, since the wrist is the furthest extremity we can reliably measure, and it moves a greater distance than the elbow or shoulder does during the tests. Once it detects motion and ensures that there hasn't been motion for another third of a second, it stops all sensor recording, gathers data from the other connected sensors, and starts feature extraction.

Despite being able to record the 3D positions of the user's joints, the Kinect has several limitations that need to be addressed to record movements such as forearm supination, pronation, and wrist dorsiflexion. The Kinect cannot perform reliable information about the orientation of the hand or fingers, which is required for a number of tests.

3.2 Wired Glove

The DG5-VHand Glove 3.0 is another sensor that we use to overcome the limitations of the Kinect. It is worn as a glove, and has flexion sensors in each of the fingers to detect how much each of the fingers can bend. There is also an IMU on the glove that is located on the back of the hand. The IMU contains an accelerometer, gyroscope, and magnetometer. This sensor allows us to support tests that would not be supported by the Kinect, such as tests requiring forearm supination and pronation. It also allows us to record movement data for tests already supported by the Kinect, such as speed or smoothness of the movement. This information is used to create extra features to aid classification of the movement. However, the main reason we use the glove is to support the ability to test finger movement and strength. The FMA contains 5 different movements that test different grasps, along with an additional two movements that test finger range of motion.

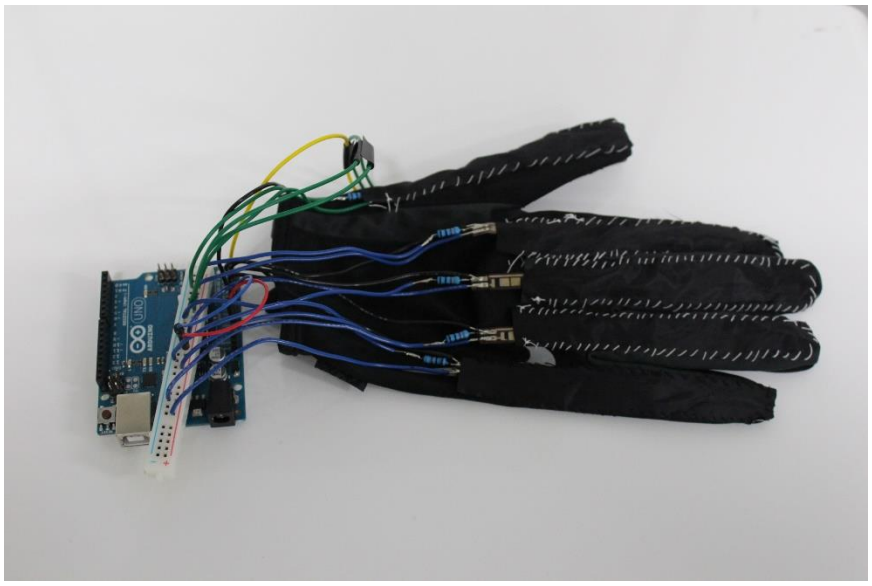
3.3 Pressure Sensor

A third sensor that we use for two of our tests (grasp 2 and grasp 3) is a simple pressure sensor connected to an Arduino Uno. We used this sensor to measure the gripping strength between the index finger and thumb for these two tests. This sensor is required because the wired glove cannot detect any information about finger strength.

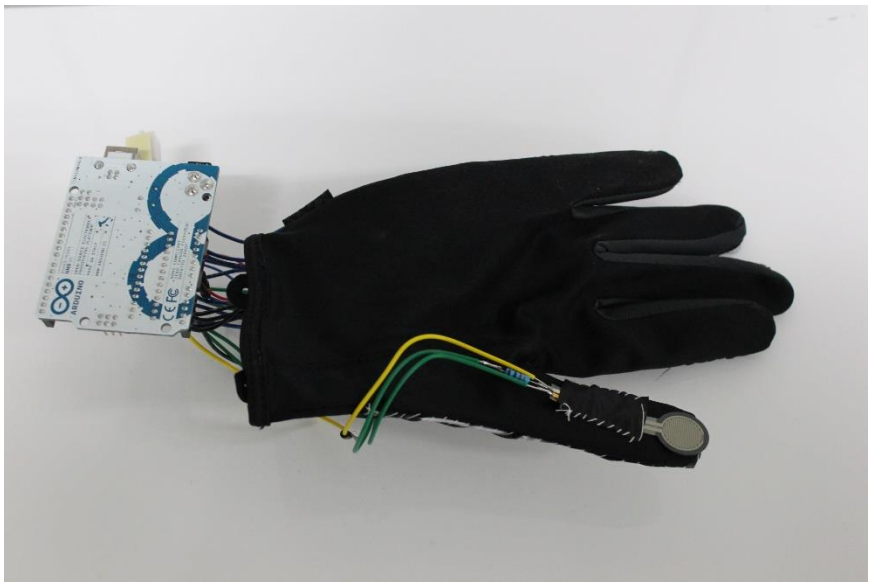
3.4 Additional Sensors

Of the sensors we added support for, there were two candidate sensors that were not included in the final system design. The first sensor was a 9 degree of freedom IMU called the Shimmer 3, shown in Figure 5. It can be worn around the wrist to measure arm movement. We excluded it from the final design because the DG5-VHand Glove 3.0 provided a better range of IMU capabilities. The second sensor was a custom-made data glove sensor that we created with an Arduino Uno, 5 flexion sensors, and a pressure sensor (shown in Figure 6). The pressure sensor was located on the pad of the thumb to allow coverage for the Grasp II and Grasp III tests. The flexion sensors could measure the degree to which each finger was bent. This sensor eliminates the need for a separate pressure sensor. However, we replaced it with the VHand Glove because the VHand Glove efficiently integrates an IMU into its design. Using the combined IMU and data glove allowed us to achieve wide test coverage with a single sensor.

Figure 5: The Shimmer 3.0 sensor. It is a 9 degree of freedom IMU that can be attached to the body via a wrist strap.



(a)



(b)

Figure 6: Our custom-made data glove sensor, built with an Arduino Uno. It includes a pressure sensor on the thumb, allowing us to support the grasping tests without needing a separate pressure sensor.

3.5 Data Aggregation

Each sensor is driven by an application that records the data from the sensors and sends them to the main application when requested. As the user selects a test movement to perform in the main application, he/she starts recording with a voice command. When the recording starts, the sensors start filling a buffer of sensor values. As data is being recorded, the main application analyzes the 3D joint position data to detect when the movement has stopped, at which point it gathers data from the other sensors and stops recording. If the user is unable to move during the recording window, the recording will time out after 5 seconds and gather the data at that time.

Communication between the main application, the sensors, and the classifier takes place through sockets, making the components of this modular system easy to swap out. This has the benefit of making it easy to include additional sensors. To include another sensor, a new program is created for the new sensor following the template being used by the sensors that are already implemented. The program must be able to take commands from the main application, to record data, and to send the recorded data back to the main application. The main application can then simply add the additional sensor to the list of sensors it connects to.

IV. MACHINE LEARNING

Classification of different types of movements is very easy for us humans to distinguish, but it can be somewhat difficult to articulate exactly what the distinguishing factors are, or to decide which factors are more important than others. Machine learning algorithms provide solutions to this problem by “learning” to recognize patterns from training data. Based on what they learn, they can then classify future examples. The two approaches used in this thesis approach the classification problem in different ways. SVM uses a statistical approach to classification, while BNN is modelled after a brain, with a network of neurons that feed inputs to each other to eventually produce an output. These two approaches were chosen because they have been proven to be most effective for pattern classification of gestures from sensor data by studies such as [17]. Other approaches, such as Naïve Bayes and Decision Tree Learning seem to perform with lower accuracy.

As our SVM classifier, we used LIBSVM, a library for SVMs [22]. To use LIBSVM, training data with known classifications must be formatted into a text file and used to train a SVM classifier. After that, LIBSVM can classify further test instances with unknown classifications by comparing the test instance with the training instances and finding the best match. Similarly, BNNs must be trained before they can be tested. The main challenges with these machine learning algorithms are to decide what features to extract from the data and to establish the best parameters for training.

4.1 SVM Overview

SVM is a useful method for classifying both linear and non-linear data. When given training instances as input, SVM tries to linearly separate the data, which is not usually possible without some pre-processing on the feature set. To pre-process the feature set, SVM maps it non-linearly to a higher dimension, which then allows for linear separation. This mapping is known as the kernel function, which is usually linear, polynomial, sigmoid, or based on the Radial Basis Function. At this point, SVM constructs the linear hyperplane that separates the different classes of training instances, which ideally has the

largest possible distance from the nearest training data points. This distance is called the margin, while the nearest training data points are called the support vectors. A simplified illustration of the mapping process and the resulting hyperplane can be seen in Figure 7. After defining this hyperplane, new testing instances can be classified by mapping it to the same dimensions as the training data was mapped to. Its classification is determined by its position relative to the hyperplane created in the training phase.

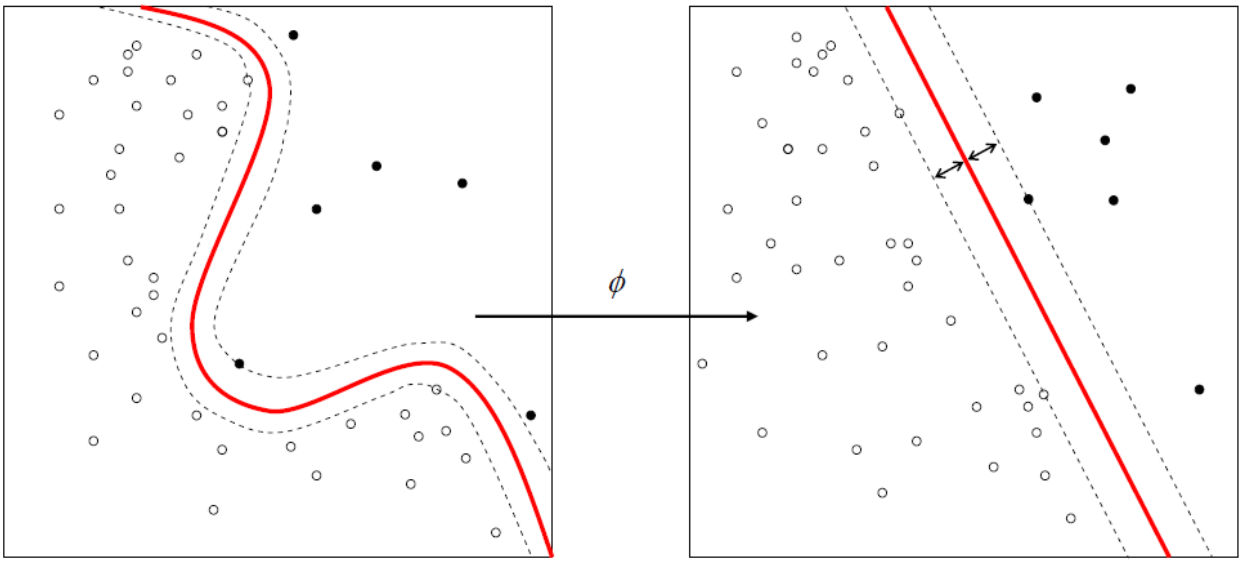


Figure 7: Pre-processing the feature set by mapping it to a higher dimension allows for linear separation of data. [23]

4.2 BNN Overview

Animal brains are very good at classification, yet they are too complex for us to understand. Therefore, artificial neural networks were created to simulate the most basic building blocks of the brain: the neuron. The perceptron, which is a model of a neuron, consists of several inputs, which are summed and compared to a threshold value to determine if it should output a value of 0 or 1. However, instead of simply deciding whether a perceptron should output 0 or 1, it can output the result of a sigmoid function, which returns a value *between* 0 and 1. In addition to this, each input for each perceptron can have its own weight applied to it. The output of the perceptron can also have a bias added to it. An illustration of a perceptron is shown in Figure 8. With this setup, Backpropagation is an algorithm that can be given training data to compute the optimal weight and bias values of the neural network to classify future test instances.

Backpropagation uses gradient descent to train artificial neural networks by calculating the output of a training instance, comparing it to the expected output value, and adjusting the weights and biases of the neural network. Gradient descent is a method for finding the local minimum of a function. In the case of training a BNN, it is used to find the local minimum of the error of the network, which is represented as the following function, where *t* is the target (expected) output and *y* is the actual output of the network.

$$E = \frac{1}{2}(t - y)^2$$

Gradient descent is performed repeatedly to adjust the weights and biases of each perceptron such that the output of the network becomes more similar to the target output specified by the training instance (by finding the local minimum of the error function). The overall process involves finding the error as the root mean square of the difference between the BNN output and expected output. Given the error, gradient descent is used to adjust the weight and bias values of the BNN to lower that error. This is done repeatedly as an iterative process until the error is below a specified threshold.

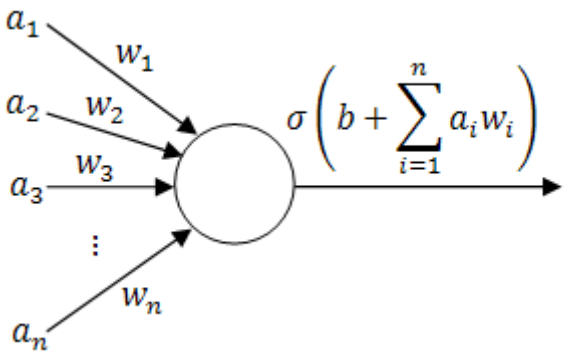


Figure 8: A perceptron that takes several inputs, a_n, with weights w_n, and gives an output as the sum of the weighted inputs passed through the sigmoid function. [24]

4.3 Feature Extraction

We used the same feature extraction methods for both SVM and BNN with the exception of scaling our data for BNN. After receiving raw sensor data from the sensors, the system extracts usable data that can be used to differentiate between different levels of limb functionality. Since every test is a different movement, they have their own feature extraction routines and SVM classifiers. This is because each movement has different requirements for what should be observed to distinguish between levels of functionality. For example, wrist supination and pronation have to be measured by checking the range of wrist rotation while elbow flexion has to be measured by checking the angle of the elbow joint during the movement.

When the user performs a test and the main system collects the sensor data from the movement, it executes the feature extraction function for that test. Most of the tests need to process Kinect skeleton frames to generate features. Before they process the skeleton frames, they need to account for the variability in the user's movement, such as speed. We measure the speed of a movement by comparing the distance traveled by the wrist from frame to frame, since it is the furthest extremity that we can reliably measure with the Kinect. To account for speed variability and any pauses in the movements, the system uses a two-phase pre-processing routine.

During the first phase, the system compares the distance the wrist has traveled in adjacent frames, starting with the first two frames. For each pair of adjacent frames, if there is not a significant amount of movement, the second frame is removed. This has the effect of slightly speeding up the movement, and the speed difference is more noticeable for very slow movements. This step is required because many stroke patients often perform test movements with varying levels of speed throughout their full range of motion, sometimes including brief pauses along the way. After this first phase, the second phase is to set the number of remaining frames to a fixed number *n*. If the number of remaining frames is greater than *n*, then frames are removed at even intervals, having the effect of further speeding up the movement. If the number of remaining frames is less than *n*, then some frames are repeated at even intervals, having the effect of slowing the movement down. This pre-processing step is necessary to perform before feature extraction because each SVM classifier requires a fixed number of features, and the number of features the system creates is based on the number of skeleton frames. The fixed number *n* is different for some of the tests because we have found that some tests are classified more accurately with different values of *n*.

Once the feature extraction routine sets the number of skeleton frames to a fixed number, it begins gathering the actual features. The first set of features gathered is the direction that relative limbs are pointing over time. Limbs are composed of a superior joint J_s and an inferior joint J_i. The limbs relevant to the test movement are converted into 3D unit vectors, which gives information about the direction that the limb is pointing, but is not affected by the length of the limb. For every skeleton frame, the 3 elements of each unit vector for each limb are recorded as features.

The next set of features extracted from the skeleton frames is the set of angles between the relevant joints. For example, the FMA specifies that the elbow must remain at 90 degrees throughout certain movements. To calculate the angle *a* for a joint, we start with the unit vector for the superior limb $S=\langle s_0,s_1,s_2\rangle$ and for the inferior limb $I=\langle i_0,i_1,i_2\rangle$ that are adjacent to the joint. We then take the arccos of the dot product of these vectors and convert it to degrees:

$$a = \arccos[(s_0 * i_0) + (s_1 * i_1) + (s_2 * i_2)]\frac{180}{\pi}$$

Each angle for each relevant joint in every skeleton frame is recorded as a feature. An illustration of the typical unit vectors and angles extracted as features is shown in Figure 9, where the angle for the elbow joint is calculated and a unit vector is created for 3 limbs: between the neck and shoulder, between the shoulder and elbow, and between the elbow and wrist.

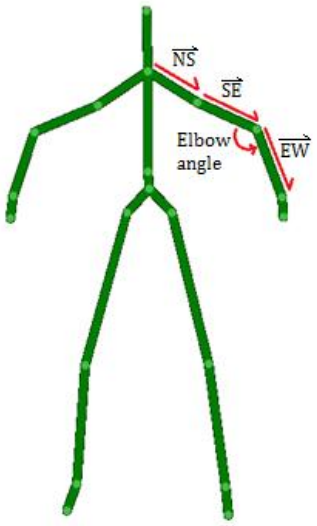
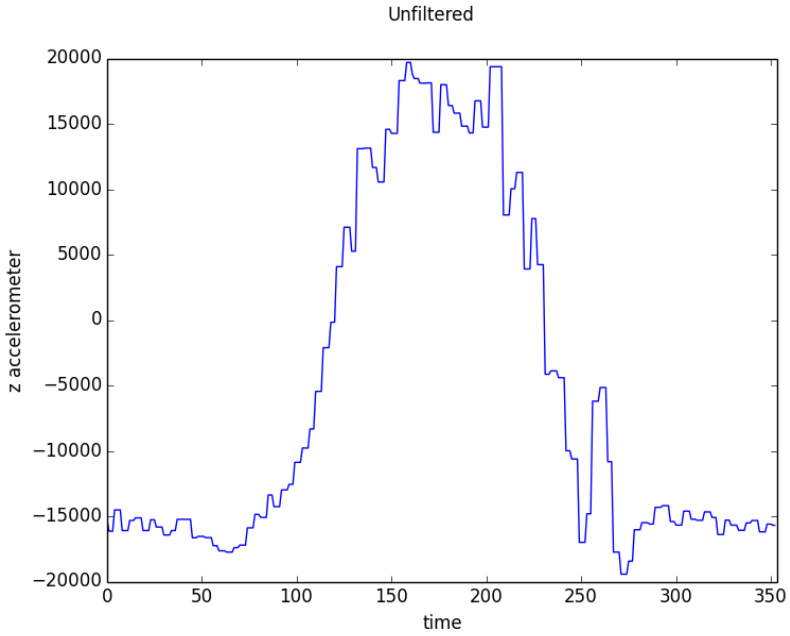
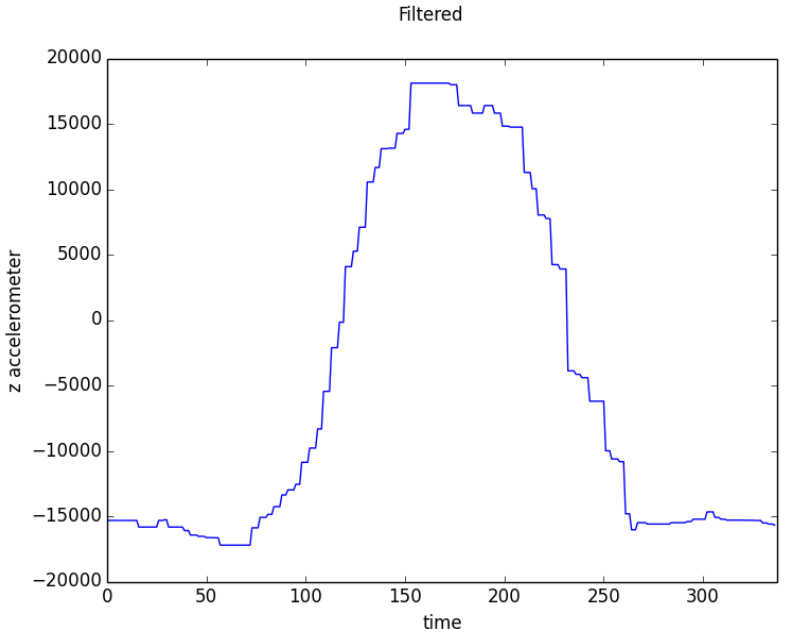


Figure 9: A depiction of the skeleton data gathered by the Kinect with the features we extracted from that data. The limbs between the neck (N), the shoulder (S), the elbow (E), and wrist (W) are converted to 3D unit vectors. The elbow angle is calculated in degrees.

Many feature extraction routines require IMU data to overcome some of the shortcomings of the Kinect. For example, the Kinect is unable to accurately distinguish twisting motions such as supination and pronation. When working with the raw accelerometer and gyroscope data, the feature extraction functions use median filtering to smooth the data. This greatly simplifies feature extraction from the data, and the effect is shown in Figure 10.

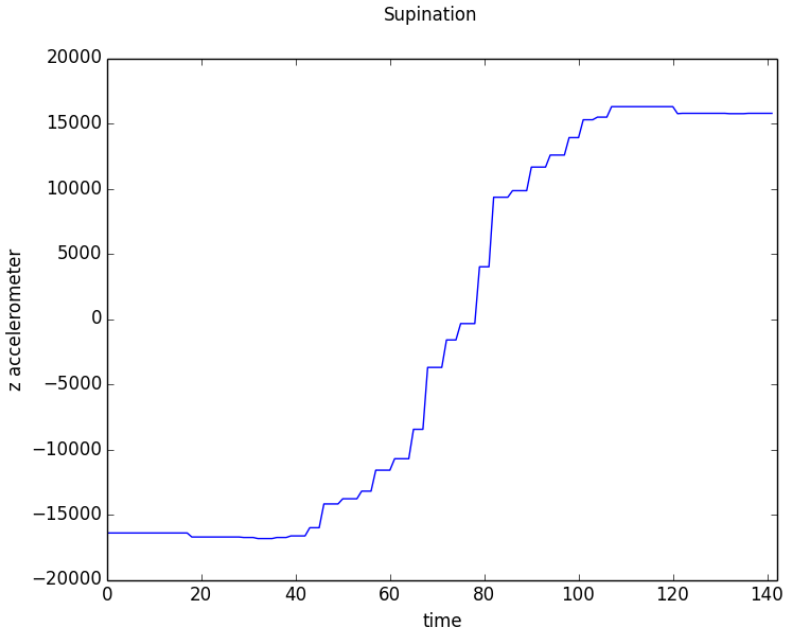


(a)

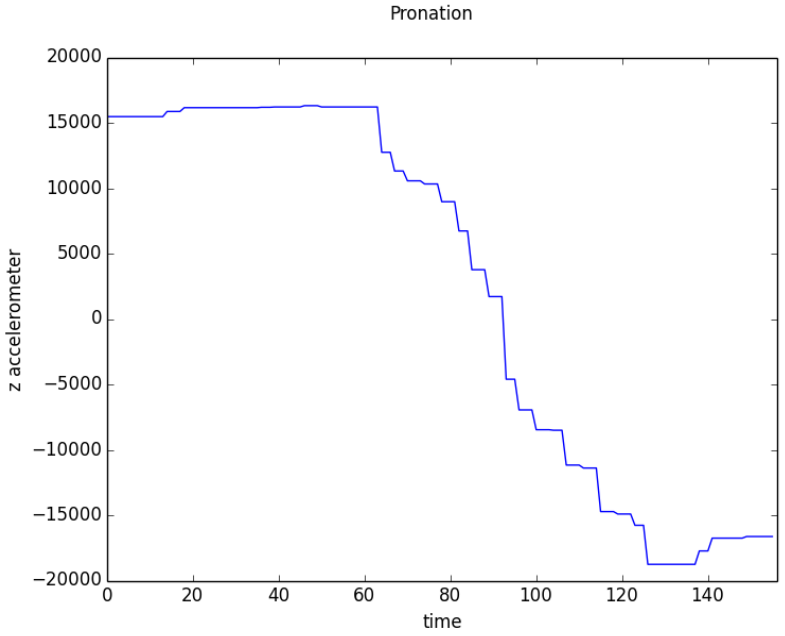


(b)

Figure 10: The effect of median filtering on a window of accelerometer readings from a supination and pronation movement. Raw sensor data is shown in (a) while the smoothed version is shown in (b).



(a)



(b)

Figure 11: Z accelerometer sensor readings of supination (a) and pronation (b) after median filtering.

For the tests that detect the range of supination or pronation, we use the sensor readings from the z accelerometer. The smoothed sensor readings for supination and pronation are illustrated in Figure 11. To determine the range of the motion, we identify windows in the sensor readings that look as they do in Figure 11 and use the difference between the maximum and minimum values. This approach is also used in other tests such as wrist circumduction and wrist dorsiflexion.

The grasp tests, which test a patient's finger functionality, are tested using the flexion sensors in the wired data glove. The values of the flexion sensors are higher the further the fingers are bent. Using this information, we take the readings from each finger and find the point where the average of all 5 finger readings are at their maximum in the set. That average is then used as the primary feature for these tests.

Some tests, such as wrist circumduction, use the "shakiness" of a movement as a factor for scoring. For these tests, we calculate a value representing the smoothness by using median filtering on the IMU data. We then sum the absolute value of the difference of each raw sensor value from the smoothed value. The resulting value is lower for smooth movements and higher for shaky movements, and is used as a feature for the tests that require it.

4.4 Training SVM

To train our SVM classifiers, we gathered data from a non-stroke volunteer. This volunteer performed each test movement in a number of different possible ways. This gave us a dataset with multiple examples of each possible score for each supported test. Given this dataset, we made a few decisions about how we would train our classifiers.

The first issue was to decide which SVM kernel function to use. The library we use, libsvm, provides 4 kernel functions: linear, polynomial, radial basis function (RBF), and sigmoid. We tested each of these kernel functions with a sample dataset and the default kernel parameters. The results, shown in Table 2, indicate that linear kernels would be best suited for our system.

Linear	Polynomial	RBF	Sigmoid
83.33%	81.25%	33.33%	39.58%

Table 2: A comparison of SVM kernel function accuracy on a sample dataset.

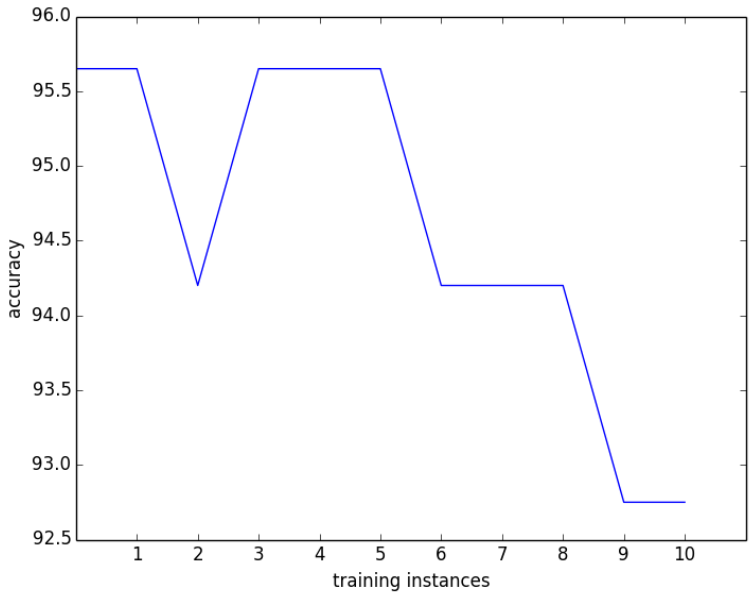
The linear kernel uses a *C* parameter to influence the size of the margin and location of the line separating the different training instances. We found that the highest accuracy is achieved when using a low value of *C*, setting it to 2^{-5}. Given this parameter, we trained each SVM classifier based on the data that we collected.

With these parameters, we ran tests to determine the best number of training examples to use. If the number of training instances were too small, the classifiers wouldn't have enough data to account for variations within the same possible classifications for each movement. If there were too many training instances, there may be an over-fitting problem where certain data points and features may have a negative effect on classification. This is counter-intuitive because logically, being shown more examples of a certain category should improve the classifier. However, this may cause the classifier to be affected by the values of certain features, resulting in a sub-optimal dividing line between data points from different classes.

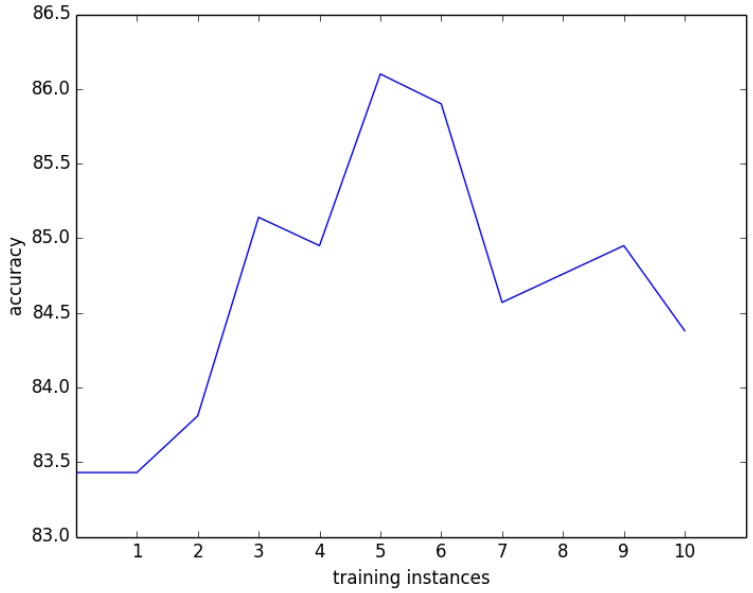
To test the best number of training examples, we recorded a volunteer performing each test 30 times. For each test, the volunteer performed 10 instances of the movement in a way that would be classified as 0, 1, and 2 each. Afterwards, the same volunteer was recorded performing each test 3 times; once for each possible classification. This gave us an 11th set of data to be used for testing.

Figure 12(a) shows our results from testing the number of training sets incrementally. The accuracy shown is the overall accuracy for all tests combined. Most of the tests showed 100% accuracy regardless of the number of training examples. The variability in accuracy came from a small number of tests. When the number of training examples was between 3 and 5, there were only 3 tests that were not classified with 100% accuracy. When the number of training examples was 6 or greater, the number of tests that

were not classified with 100% accuracy increased to 5. We repeated this test with all the data collected from volunteers with the results shown in Figure 12(b). These results indicate that 5 training sets for each class of each test is the optimal amount.



(a)



(b)

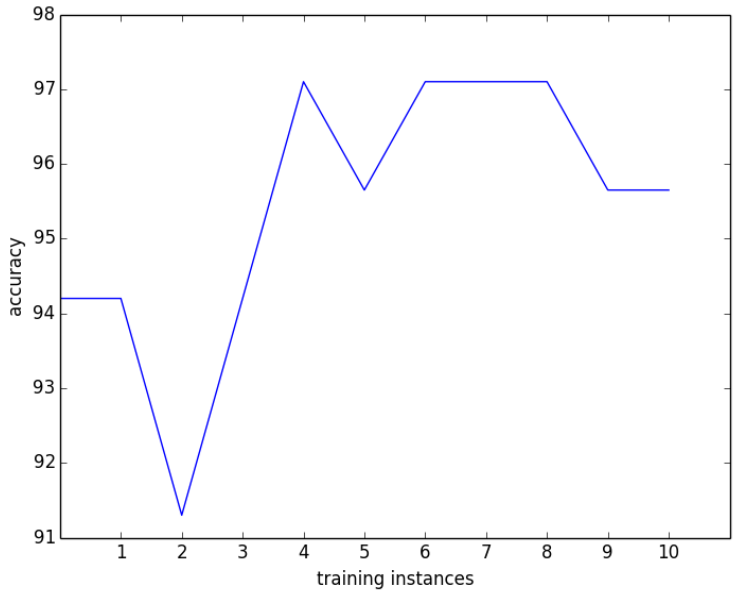
Figure 12: The accuracy of the SVM classifiers given the number of training examples for the individual who created the training data (a) and the combination of data from all volunteers (b).

4.5 Training BNN

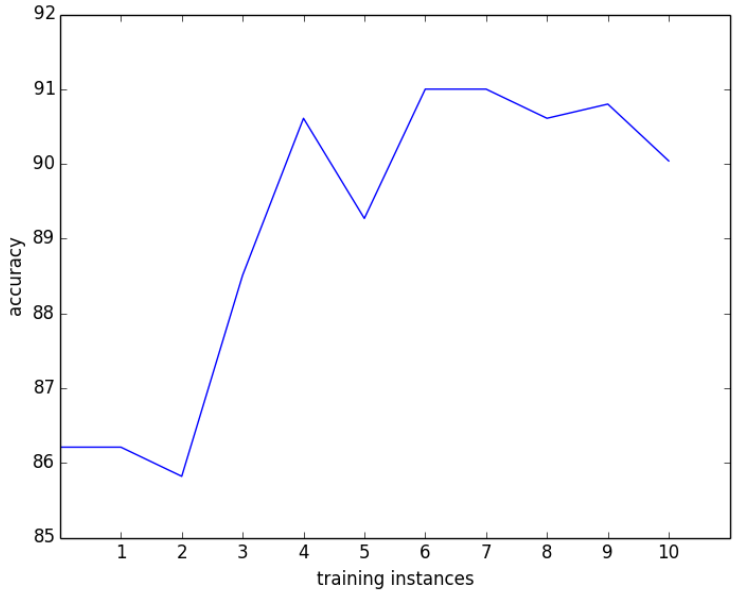
To use BNN as a classifier, we created a separate neural network for each test. The input nodes in the BNNs are the same as the inputs to the SVMs, with the exception of the data being scaled to values between 0 and 1. Therefore, the number of input nodes to each BNN is equal to the number of features for each corresponding test. The number of output nodes is 3. For the output, the classification is considered to be the number of the node with the maximum value. For example, if the first node had the highest value, the classification would result as a 0, while if the last node had the highest value, the classification would result as a 2. Each network had one hidden layer.

The Backpropagation algorithm is based on the Gradient Descent algorithm to determine the optimal weights and biases for each node in the neural network. Therefore, the main parameters to be considered are the learning rate, the learning momentum, and the epochs, or number of iterations we use to train the neural network on each training instance. The final parameter is the number of hidden nodes. After some trial and error, we found the best learning rate to be 0.5 and the best learning momentum to be 0.1. The maximum number of epochs was set to 1000. For each test, the number of hidden nodes was typically set to a value halfway between the number of input nodes and the number of output nodes, with a maximum possible value of 50 hidden nodes.

Given the training parameters, we ran tests to determine the optimal number of training instances for the BNNs. Just like SVM, BNN can be affected by over-fitting and under-fitting. Therefore we tested the effect of the number of training instances on the overall training accuracy. The results of these tests are shown in Figure 13, and indicate that the system performs best with 6 training examples from each class. Figure 13(a) shows the accuracy based on testing instances from a single individual while Figure 13(b) shows the accuracy based on testing instances from all volunteers we collected data from.



(a)



(b)

Figure 13: The accuracy of the BNN classifiers given the number of training examples for the individual who created the training data (a) and the combination of data from all volunteers (b).

V. EXPERIMENTAL RESULTS

The first step in testing our system was to assess the accuracy of our system using data collected from non-stroke volunteers. We gathered data from 8 volunteers, who were each asked to perform each test three times: once where the movement was performed faultlessly, once where it was performed partially, and once where it wasn't performed at all. This gave us a data set of test movements resulting in scores of 0, 1, and 2. With this data set, we gathered information for our feature extraction routines and the amount of training data required for our system to provide reliable results.

After determining the best number of training instances as discussed in the previous section, we tested the system using data from several volunteers. To get another measure of accuracy, we also performed v-fold cross validation on the data collected from our volunteers for both SVM and BNN. The results, shown in Table 3, indicate that our SVM classifiers are capable of a maximum accuracy of up to 96.83%, with an overall average accuracy of about 86.1%. The BNN classifiers are capable of a maximum accuracy of 96.83% and an overall accuracy of 93.12%. These results show that BNN generally has better classification accuracy than SVM. This is possibly due to the fact that BNN is not as affected by outliers in the training set as SVM is. The way we set up the outputs to the BNNs as 3 output nodes may also allow BNN to handle some slight errors better than SVM. These results are promising because they demonstrate that this movement data can feasibly be distinguished by machine learning algorithms that can be used to extend this work for stroke rehabilitation applications.

The main reason for the inaccuracies in testing our sensor data is the noise from the Kinect sensor. When initializing the Kinect, the system enables the skeleton joint coordinate smoothing features implemented in the Kinect SDK. Despite this, there is often a lot of jittering with many of the joints when recording with the Kinect. In a number of cases, the Kinect miscalculates the position of an entire arm, especially if there is any occlusion. To account for this, we included noisy data into our training sets. When there is severe occlusion, the volunteers are given instructions to adjust the direction they were facing relative to the Kinect.

Our system was tested on a laptop with an Intel Core i7 processor and 8 GB of RAM. Given these specifications, it typically took our system about 1.5 seconds to extract features from sensor data and classify them for each movement. Therefore, our system is capable of recording a movement and classifying it in real time.

Person	SVM Accuracy	SVM V-fold Accuracy	BNN Accuracy	BNN V-fold Accuracy
p_1	93.65%	90.48%	95.24%	96.83%
p_2	75.76%	86.36%	87.88%	93.94%
p_3	84.85%	89.39%	93.06%	95.83%
p_4	78.79%	84.85%	81.82%	86.36%
p_5	89.23%	90.77%	83.10%	88.73%
p_6	87.88%	87.88%	93.10%	93.10%
p_7	95.65%	89.39%	96.00%	93.33%
p_8	84.13%	96.83%	90.48%	96.83%

Table 3: An overview of the classification accuracy of our SVM and BNN classifiers for each of our non-stroke volunteers.

VI. CONCLUSION

In this thesis, we proposed a method for using sensors to automate the FMA to evaluate the upper-limb condition of post-stroke patients. We have implemented the system as a proof-of-concept to demonstrate feasibility and are in the process of receiving approval for actual stroke patient tests. Once the reliability of this system is demonstrated, it will offer an inexpensive method for automating post-stroke patient upper limb evaluation. The primary benefit of this system is that it provides a method of evaluation that can be performed remotely without the presence of a doctor. This system saves up to 30 minutes of a doctor's time for each patient that they have, providing an inexpensive, time-saving service. After testing both SVN and BNN as classifiers, our preliminary experiments on healthy volunteers have shown that our system can achieve an overall accuracy of 93.12%.

Given our results, we envision a few different ways to extend the work. With the design we created, the application presented in this paper can be extended to support more tests using more sensors. To add support for an additional test, an additional feature extraction routine should be created that specifies relevant information to the test. This routine has to specify features to be gathered from the available sensor data. Typical features include elbow and shoulder joint angles, forearm and upper arm limb directions, and hand orientation. With this approach, tests in other FMA domains can be supported, such as lower extremity motor. In addition, accuracy of the tests currently supported can be increased by specifying different features, utilizing data from an additional sensor, or using another machine learning algorithm for classification.

References

[1] Fugl-Meyer, A. R., Jääskö, L., Leyman, I., Olsson, S., & Steglind, S. (1974). The post-stroke hemiplegic patient. 1. a method for evaluation of physical performance. Scandinavian journal of rehabilitation medicine, 7(1), 13-31.

[2] Duncan, P. W., Propst, M., & Nelson, S. G. (1983). Reliability of the Fugl-Meyer assessment of sensorimotor recovery following cerebrovascular accident. Physical therapy, 63(10), 1606-1610.

[3] Gladstone, D. J., Danells, C. J., & Black, S. E. (2002). The Fugl-Meyer assessment of motor recovery after stroke: a critical review of its measurement properties. Neurorehabilitation and Neural Repair, 16(3), 232-240.

[4] Sanford, J., Moreland, J., Swanson, L. R., Stratford, P. W., & Gowland, C. (1993). Reliability of the Fugl-Meyer assessment for testing motor performance in patients following stroke. Physical therapy, 73(7), 447-454.

[5] Sullivan, K. J., Tilson, J. K., Cen, S. Y., Rose, D. K., Hershberg, J., Correa, A., ... & Duncan, P. W. (2011). Fugl-Meyer Assessment of Sensorimotor Function After Stroke Standardized Training Procedure for Clinical Practice and Clinical Trials. Stroke, 42(2), 427-432.

[6] Liu, L., Wang, D., Wong, K. L., & Wang, Y. (2011). Stroke and stroke care in China huge burden, significant workload, and a national priority. Stroke, 42(12), 3651-3654.

[7] Langan, J., DeLave, K., Phillips, L., Pangilinan, P., & Brown, S. H. (2013). Home-based telerehabilitation shows improved upper limb function in adults with chronic stroke: a pilot study. Journal of Rehabilitation Medicine, 45(2), 217-220.

[8] Holden, M. K., Dyar, T. A., & Dayan-Cimadoro, L. (2007). Telerehabilitation using a virtual environment improves upper extremity function in patients with stroke. Neural Systems and Rehabilitation Engineering, IEEE Transactions on, 15(1), 36-42.

[9] Webster, D., & Celik, O. (2014, February). Experimental evaluation of Microsoft Kinect's accuracy and capture rate for stroke rehabilitation applications. In Haptics Symposium (HAPTICS), 2014 IEEE (pp. 455-460). IEEE.

[10] LaBelle, K. (2011). Evaluation of Kinect joint tracking for clinical and in-home stroke rehabilitation tools. Undergraduate Thesis, University of Notre Dame.

[11] Pastor, I., Hayes, H. A., & Bamberg, S. J. (2012, August). A feasibility study of an upper limb rehabilitation system using kinect and computer games. In Engineering in Medicine and Biology Society (EMBC), 2012 Annual International Conference of the IEEE (pp. 1286-1289). IEEE.

[12] Chongyang, Y., Zixi, W., & Linhong, J. (2008, May). A research on the feasibility of automatic Fugl-Meyer assessment for upper limb based on rehabilitation devices. In Proceedings of the 2nd International Convention on Rehabilitation Engineering & Assistive Technology (pp. 126-129). Singapore Therapeutic, Assistive & Rehabilitative Technologies (START) Centre.

[13] Huang, M. C., Xu, W., Su, Y., Lange, B., Chang, C. Y., & Sarrafzadeh, M. (2012, June). Smartglove for upper extremities rehabilitative gaming assessment. In Proceedings of the 5th International Conference on Pervasive Technologies Related to Assistive Environments (p. 20). ACM.

[14] Yeh, S. C., Lee, S. H., Wang, J. C., Chen, S., Chen, Y. T., Yang, Y. Y., ... & Hung, Y. P. (2012, October). Virtual reality for post-stroke shoulder-arm motor rehabilitation: Training system & assessment method. In e-Health Networking, Applications and Services (Healthcom), 2012 IEEE 14th International Conference on (pp. 190-195). IEEE.

[15] Wang, J., Yu, L., Wang, J., Guo, L., Gu, X., & Fang, Q. (2014, April). Automated Fugl-Meyer Assessment using SVR model. In Bioelectronics and Bioinformatics (ISBB), 2014 IEEE International Symposium on (pp. 1-4). IEEE.

[16] Olesh, E. V., Yakovenko, S., & Gritsenko, V. (2014). Automated Assessment of Upper Extremity Movement Impairment due to Stroke. PloS one, 9(8), e104487.

[17] Patsadu, O., Nukoolkit, C., & Watanapa, B. (2012, May). Human gesture recognition using Kinect camera. In Computer Science and Software Engineering (JCSSE), 2012 International Joint Conference on (pp. 28-32). IEEE.

[18] Biswas, K. K., & Basu, S. K. (2011, December). Gesture Recognition using Microsoft Kinect®. In Automation, Robotics and Applications (ICARA), 2011 5th International Conference on (pp. 100-103). IEEE.

[19] Nirjon, S., Greenwood, C., Torres, C., Zhou, S., Stankovic, J. A., Yoon, H. J., ... & Son, S. H. (2014, March). Kintense: A robust, accurate, real-time and evolving system for detecting aggressive actions from streaming 3D skeleton data. In Pervasive Computing and Communications (PerCom), 2014 IEEE International Conference on (pp. 2-10). IEEE.

[20] Raptis, M., Kirovski, D., & Hoppe, H. (2011, August). Real-time classification of dance gestures from skeleton animation. In Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (pp. 147-156). ACM.

[21] Kinect SDK. Available: http://www.microsoft.com/en-us/kinectforwindows

[22] Chang, C. C., & Lin, C. J. (2011). LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology (TIST), 2(3), 27.

[23] Alisneaky, “Kernel Machines are used to compute a non-linearly seperable functions into a higher dimension linearly seperable function.”, Wikimedia Commons, 2011

[24] Manfred Zabarauska, “Backpropagation Tutorial”, Manfred Zabarauskas’ Blog, 2011

요약문

Supervised 기계학습을 이용한 Fugl-Meyer 평가의 자동화

치료 후 반신불수인 뇌졸증 환자의 호전을 평가하는 것은 재활에 있어서 중요한 사항이다. 이것은 또한 임상 실험 동안 수행하는 필수적인 단계이다. Fugl-Meyer Assessment (FMA)는 뇌졸증 환자들의 신체기능 장애를 측정하는 방법들 중 가장 널리 알려졌을 뿐만 아니라 널리 이용 되는 방법이다. 우리는 환자의 동작을 관찰하는 센서로부터 데이터를 모음으로써 팔 부분의 FMA 를 자동화하는 방법을 제안한다. 특징들은 데이터로부터 추출되고 기계학습 알고리즘에 의해 처리된다. 기계학습 알고리즘으로부터 얻은 출력은 환자의 위쪽 팔의 기능을 수치화하는데 사용되는 값을 반환한다. 우리 시스템에서는 Support Vector Machines(SVM)과 Backpropagation Neural Networks (BNN) 이라는 기계학습 알고리즘을 사용한다. 이 시스템은 저렴하고 자동화된 뇌졸증 환자 평가를 하도록 도와 줄 것이다. 게다가 뇌졸증 연구원과 의사들에게 시간을 줄여주는 서비스를 제공하여 의사가 환자당 30 분을 절약 할 수 있을 것이다.

핵심어: stroke assessment, Fugl-Meyer Assessment, machine learning, Support Vector Machine, Kinect

Acknowledgement

This research was supported in part by the DGIST Research and Development Program of the Ministry of Science, ICT and Future Planning of Korea (CPS Global Center).

Curriculum Vitae

Name: Paul Otten

Birth Date: August 7th, 1990

Education

8/2008 – 12/2012	George Mason University
	B.S. Computer Science
2/2013 – 2/2015	Daegu Gyeongbuk Institute of Science and Technology
	M.S. Computer Science

Work Experience

1/2009 – 6/2011	George Mason University
	ITU Support Center Analyst / Lead Analyst
6/2011 – 6/2012	IBM – International Business Machines Corp.
	Software Developer (Co-op/Internship)

Honors and Awards

2/2007	CompTIA A+ Certification
8/2010	Linux Professional Institute Level 1 Certification