**ABSTRACT**

The purpose of this project is to identify human activities while using cell phones via mobile sensor data. Around 2.1 billion people use smartphones in present day each smartphones equipped with different sets of sensor which are able to perform various actions if placed on right track. So here through this paper we present our recent work on human activity detection using these sensors and various learning algorithms. The proposed human activity detection system recognizes 6 different class of human activities like walking, running, siting etc. In our system with the help of embedded sensors like tri-axial accelerometer, tri-axial linear accelerometer are used for motion data collection. With these collected data now used in building prediction model which use two stage data analysis approach i.e. Short period statistical analysis (max, min, mean, and standard deviation) and long period data analysis using machine learning. The system is implemented in an Android smart phone platform.

**CHAPTER ONE**

**INTRODUCTION**

Being able to automatically infer activity of human that they are performing will be a great assist on different applications like personal assistive robotics, medical purpose and various other. For example, if there is an app that can keep track of human activity like their finger movement or they are standing sitting or walking will be a great assist in the field of medical science no separate nurse has to be there to keep track of the patient vitals. The activity that happen in daily life are not in structured format; the data that obtained from personal daily activity of person are unsupervised or unstructured .they consist of anomalies as well as data generate from each person are different in pattern no two person follow similar pattern in their daily life. Thereby in simple fact they are just raw facts like data in computer science which needs to be processed in order to use and generate pattern from them. And due to this unstructured, anomalies consisting and different data pattern creates confusion and creates problem in detecting actual activity of a person. In this work, we are interested in processing these unstructured data and creates a separate pattern which will ease in detecting human activity such as sitting walking standing etc.

Most prior works in human activity detection have focused on activity detection from still images or from 2D videos (or use of RFID sensors placed on humans and objects), 3D videos and expansive cameras and sensors. The use of RFID tags is generally too intrusive because it requires a placement of RFID tags on the people. Estimating the human behavior is the primary focus of these works, and they consider actions taking place over shorter time scales. Having access to accelerometer and gyroscope of smart phones, enables us to robustly estimate human poses and use this information for learning complex human activities

In this work we are going to use inexpensive easily available mobile sensors like gyroscope and accelerometer that is present in every smart phone. Our focus in this work is to obtain a descriptive labeling of the complex human activities that take place over long time scales and consist of a long sequence of sub-activities, such as walking running sleeping etc. For example, a user is moving suddenly changing his activity and started sitting on ground our system gives a labeling of their activity what they are doing in real time. This show how challenging is the task to take care of people's activity and provided labeling of their activity on smart phones in real time.

In most previous work detection of activity, recognizing them and giving labeling of the activity is taken as 3 distinct tasks. Only in some works now shows detection and recognition of activity as a single task; these works have shown that modeling mutual context is beneficial. The key idea in our work is to note that, in activity detection, it is sometimes more informative to know how an object is doing rather than knowing what the object is (i.e., the object category). For example, detecting person in a room is less better than detecting what person is doing is far more beneficial. We propose a method to learn human activities by modeling the sub-activities and affordances of the objects, how they change over time, and how they relate to each other.

We use the gyroscope and acclerometer present on smart phone to get the users horizontal and radial axis then that data is processed found their mean standard deveiation median and far more more mathmetical calculation is done. Then with obtained result is then modeled using knn and svm algrothim.and generated our model to detect human activity.In this work, we perform sampling of several segmentations, and consider labelings over these temporal segments as latent variables in our learning algorithm.

We first demonstrate significant improvement over previous work on WCIM Activity Dataset . We then contribute a new dataset from real time modeling of our system. In extensive experiments, we show that our approach outperforms the baselines in both the tasks of activity as well as affordance detection. We achieved an accuracy of 93%.

**1.1 Background:**

Physical activity is well known by the general public to be crucial for leading a healthy life. Precise recording of the conducted activities is an essential requirement of their research.

This data can be used to design and construct activity recognition systems. These systems allow physicians to check the recovery development of their patients automatically and constantly(da Costa Cachucho et al., 2011). Another rising concern is the sedentary life many people live, due to the shift in lifestyle occurring in the modern world, where work and leisure tend to be less physically demanding(Gyllensten, 2010). Several reports have already found links between common diseases and physical inactivity (Preece et al., 2009). Thus, activity recognition can be used by recommender systems to help the users track their daily physical activity and promote them to increase their activity level.

With the recent progress in wearable technology, unobtrusive and mobile activity recognition has become reasonable. With this technology, devices like smartphones and smartwatches are widely available, hosting a wide range of built-in sensors, at the same time, providing a large amount of computation power. Overall, the technological tools exist to develop a mobile, unobtrusive and accurate physical activity recognition system. Therefore, the realization of recognizing the individuals’ physical activities while performing their daily routine has become feasible. So far, no-one has investigated the usage of light-weight devices for recognizing human activities.

**1.2 PROBLEM STATEMENT**

People perform different kinds of activities in daily life. Everybody has their own pattern to complete their daily lives. There are different sectors in human activities and the problems related to it. Some of these includes the intruders in a house, breaking traffic rules, employing nurses just to record the movements of a patient etc. When there is an intruder in a house, there is an unusual human activity going on other than a regular pattern. There is a different activity pattern in a traffic condition and different kind when traffic rules break. When a patient needs to be kept under observation, it is surely by reading the activity of a patient. These are some common problems which deals with the human activities and can be solved with human activity recognition module

**1.3 OBJECTIVES**

The main objective through our human activity recognition project is mainly related with the health diagnosis of a person. Inspite of that we divide our objective in two parts;

**1.3.1 General Objective:**

To design a system that collects the data from smart phone sensors and uses them to determine the activity of the person which could be used in a game development and medical diagnosis.

**1.3.2 Specific Objective:**

To prepare complete new dataset for human activity detection.

To prepare modules for medical diagnosis

**CHAPTER TWO**

**LITERATURE SURVEY**

Most successful Human-Activity Recognition (HAR) research has focused on the recognition of relatively simple activities (e.g., sitting or walking) rather than more complex activities (e.g., cooking or cleaning). Very early work in the field used data collected from sensors placed on different locations of the body. These data were straightforward to analyze, but inconvenient to collect. As a result, video recordings soon became an area of significant HAR research. Scientists have used camera recordings to recognize both full-body movements and the movement of smaller body parts, e.g., hand-gestures. However, video recordings are only slightly less inconvenient to collect than data from sensors attached to the body[.6] The data set that we employ has been used previously to test different human activity recognition methods. Much of the previous work has been focused on the development of ‘hardware friendly’ methods, which can be performed on a smartphone. The primary method resulting from this research is a hardware-friendly implementation of support vector machines (SVMs). It utilizes six SVM classifiers, in ‘one vs. all’ style (described later) to predict which observations fall into each activity category. This method obtains about 90% accuracy.

In Brezmes et al, the authors implemented a real-time classification system for some basic human activities from accelerometer data, including walking, climbing-down stairs, climbing-up stairs, sitting down, standing up, and falling. Their monitoring system was decentralized, which meant no server processing data are involved. Kwapisz et al. Built another HAR system which can identify six human activities. The data were collected from smartphone embedded accelerometers placed in the subjects’ front pants leg pockets. Chiang et al. proposed a portable activity pattern recognition system to identify physical activities. Accelerometer and GPS data were collected and four classifiers were tested. Anjum et al. proposed a smartphone application using accelerometer and gyroscope sensors as well as GPS signals to detect seven activities. C4.5 decision tree classifier was reported to yield the best performance. All the systems reviewed above are based on the supervised learning approach for activity detection. Kwon et al. examined the unsupervised learning method for human activity recognition based on smartphone sensors.[7] Absent concerns about hardware, it appears that the most effective method tried to-date is an unconstrained (i.e., non-hardware-friendly) version of the one vs. all SVM classifier, with a Gaussian kernel.c The method yields about 96% accuracy and serves as our performance benchmar.

Bao et al. [8] developed algorithms to detect physical activities from everyday tasks, and observed that while some activities are classified more accurately with subject-independent training data, others require subject-specific training data. This suggests that multiple sensors aid in recognition because conjunctions in acceleration feature values can help to identify many activities. Mannini et al. [9] analyzed activity recognition for ambulatory monitoring and pervasive computing systems, where classification of human motion is analyzed, with a focus on the computational cost employed for this purpose. The group employed naive bayes, hidden markov models and support vector machines, amongst other algorithms. Moreover, prior work with the data set that we utilize has employed all 561 features, rather than paring down the features to only those that enhance predictive accuracy. We attempt to remedy both of these gaps in the literature, in the hopes that we might obtain better algorithm performance.

Past work focused on the use of multiple accelerometers placed on several parts of the user’s body. Bao and Intille used five bi-axial accelerometers distributed across the user’s body. They tested their approach with data of twenty users.[22] (Bao and Intille, 2004). Krishnan et al. used two accelerometers to recognize five activities [25] (Krishnan et al., 2008). They collected data from only three users. Parkka et. al. created a system using twenty different types of sensors in order to recognize activities such as football, croquet, and using the toilet [23] (Parkka et al., 2006). Subramayana et. al. addressed normal daily activities by using data not only from a tri-axial accelerometer, but from micro-phones, temperature sensors and barometric pressure sensors as well [24] (Subramanya et al., 2012). These systems using multiple accelerometers and other sensors were capable of identifying a wide range of activities. However, they are not practical as they involve the user wearing multiple sensors distributed across his body.

Other studies focused on the use of a single accelerometer for activity recognition. Long et al. placed a tri-axial accelerometer worn at the user’s waist, recognizing walking, jogging, running, cycling, and sports of twentyfour users [27](Long et al., 2009). Lee et. al. used a single accelerometer attached to the left waists of only five users [26](Lee, 2009). However, all of these studies used devices specifically made for research purposes. Several investigations have considered the use of widely available mobile devices. Ravi et. al. collected data from only two users wearing a single accelerometer-based device and then transmitted this data to the phone carried by the user (Ravi et al., 2005). Lester et. al. used accelerometer data from a small set of users along with audio and barometric sensor data to recognize eight daily activities (Lester et al., 2006). However, the data was generated using distinct accelerometer-based devices worn by the user and then sent to the phone for storage. Some studies took advantage of the sensors incorporated into the phones themselves. Yang developed an activity recognition system using a smartphone to distinguish between various activities [28](Yang, 2009).

However, stair climbing was not considered and their system was trained and tested using data from only four users. Brezmes et. al. developed a real-time system for recognizing six user activities (Brezmes et al., 2009). In their system, an activity recognition model is trained for each user, i.e., there is no universal model that can be applied to new users for whom no training data exists. Bayat et al. gathered acceleration data from only four participants, performing six activities. (Bayat et al., 2014) Shoaib et al. evaluated different classifiers by collecting data of smartphone accelerometer, gyroscope, and magnetometer for four subjects, perfoming six actvities. (Shoaib et al., 2013)

**CHAPTER THREE**

**METHODOLOGY**

**3.1 DATASET**

The data set on which we test our methods is courtesy of the UCI Machine Learning Repository.[2] For the construction of original dataset, an experiment was carried out with 30 participants, having each person wear a Samsung Galaxy S2 smartphone containing an accelerometer and a gyroscope, while performing the six activities I.e. Laying, Sitting, Standing, Walking, Walking Downstairs (WD), and Walking Upstairs (WU). There are, altogether, 561 features in the data set; these are primarily based on the 3-axial linear acceleration and 3-axial angular velocity. Each observation of the data set contains information regarding one sample window – a segment of continuous time that a particular person spent doing a particular activity. The signals from those windows were processed using various filtering techniques. The features in the data set are summaries of those processed time-domain signals (captured at a constant rate of 50Hz). For example, the feature set includes the mean, standard deviation, and skewness of the gravity acceleration signal.

In preparation for our modeling, The dataset has been split into 70% training and 30% test data, with 21 of 30 participants in the train data and the remaining 9 participants in the test data. The disjoint nature of the training and test split is important to consider; an effective model at recognizing activities should be able to predict the activities of new individuals. Since each study participant walks, stands and generally performs activities with differences in his or her movements, testing the performance of the model on individuals not in the training data is critical. While a model trained and tested on the same set of individuals could perform better, this would not meet the objective of our project.

**3.2 Data Visualization**

To capture the structure of our data, and better understand the distinctions between the categories of our dataset, we implemented two well-known algorithms: principal component analysis (PCA) and t-distributed stochastic neighbor embedding analysis (t-SNE). Below, we outline these the two methods that we use to extract sets of features from our data.

**3.2.1 Principal Components Analysis:**

PCA is a technique for reducing the number of dimensions in a dataset whilst retaining most information. It is using the correlation between some dimensions and tries to provide a minimum number of variables that keeps the maximum amount of variation or information about how the original data is distributed.[12]

Principal Components Analysis (PCA) attempts to identify the directions in feature space–called ‘principal components’ or ‘PCs’–along which the data vary the most.[10] If each column of the data matrix (X) is centered and scaled, the PCs are simply the (normalized) eigenvectors of the empirical covariance matrix

Σ = 1 m Σ m I =1x (i) x (i) T

For a d-dimensional summary of the data that captures as much variance as possible, we represent the data in the basis of the first d PCs, {u1, ..., ud}.[ 10 ] That is, we compute the entries of Z as

Zij = u T j x (i) for j = 1, ..., d and i = 1, ..., m

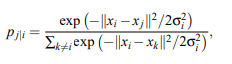
The entries are referred to as ‘scores’ or ‘PC scores’. The j-th column of Z – the ‘j-th PC score vector’ – is the projection of the data onto the j-th PC.[10].We obtain the PCs of the train-val data, after centering and scaling the original features. We then compute the PC-based features that we will use as model input by representing the train-val data and the test data in the basis of those PCs. In Figure 1, we graph the first and second PC scores of each observation in the data set, colored by activity. As is clear, even the first and second PC scores separate some activities from others.

**3.2.2 T-Distributed Stochastic Neighbouring Entities (t-SNE)**

t-Distributed Stochastic Neighbor Embedding ([t-SNE](http://lvdmaaten.github.io/tsne/)) is another technique for dimensionality reduction and is particularly well suited for the visualization of high-dimensional datasets.

*“t-Distributed stochastic neighbor embedding (t-SNE) minimizes the divergence between two distributions: a distribution that measures pairwise similarities of the input objects and a distribution that measures pairwise similarities of the corresponding low-dimensional points in the embedding”.[12]*

Stochastic Neighbor Embedding (SNE) starts by converting the high-dimensional Euclidean distances between data points into conditional probabilities that represent similarities. The similarity of data point x j to data point xi is the conditional probability, p j|i , that xi would pick x j as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at xi . For nearby data points, p j|i is relatively high, whereas for widely separated data points, p j|i will be almost infinitesimal (for reasonable values of the variance of the Gaussian, σi). Mathematically, the conditional probability pj|i is given by[13]



where σi is the variance of the Gaussian that is centered on data-point xi

**3.2.3 CLASSIFICATION ALGORITHMS**

As mentioned above, our main objective is to construct a highly accurate classifier that generalizes well on data from new individuals. For this purpose, we have tested the performance of different classifiers, and assessed why some models performed well while others performed poorly. We involved following steps for each feature engineering method and classification algorithm:

1. Build models {MV1, MV 2 , ..., MV N } on the training data, using the first {1, 2, ..., N} score vectors.

2. Use each of the models created in Step 1 to predict the classes of observations in the validation data; calculate each model’s misclassification errors. Identify the model, MV q , with the lowest error.

3. Using the first q score vectors, build MF on the train-val data.

4. Use MF to predict the classes of observations in the test data and calculate the misclassification error.

Steps 1 and 2 identify the number of score vectors (q) expected to produce the lowest misclassification error. Steps 3 and 4 provide an unbiased estimate of the algorithm’s error, when using the first q score vectors as features.

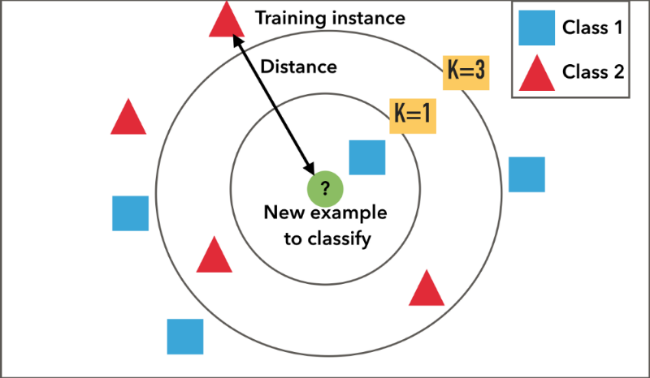
Below, we describe the classification algorithms to which we apply the above procedure

**3.2.3.1 k-Nearest Neighbors (kNN)**

k-Nearest-Neighbors (kNN) classifies a point based on the other points that are nearest to it. The predicted probability that a query point falls into class j is proportional to the number of the k training points nearest to the query point that are in class j. Mathematically, the probability that a query point falls into class j is



where N is the set of indicies of the k points closest to x (i) .



*Example of k-NN classification. The test sample (green circle) should be classified either to the first class of blue squares or to the second class of red triangles. If k = 3 (solid line circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If k = 5 (dashed line circle) it is assigned to the first class (3 squares vs. 2 triangles inside the outer circle).[14]*

As explained above we build different model on different score vectors which is shown in fig below:

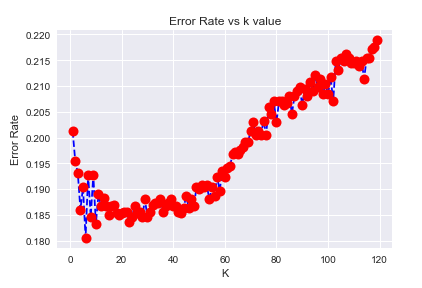


Fig: different error value at different score vector

From above fig we found that the lowest mis-classification error is at score vector 6.Now with this score vector we build our final model MF and predict the classes of observations on test-data by creating confusion matrix of them which is shown in fig below:

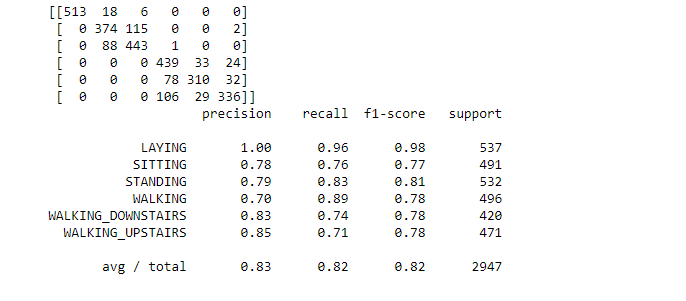


Fig : prediction of classes observation on test-data using kNN algrothim

**3.2.3.2 Support Vector Machines**

some clusters fully overlap while other clusters only partially overlap, dependent on how the corresponding activities were performed. Therefore, we would expect maximizing margins when separating these activities to result in good performance. We chose to implement SVMs using a one-vs-one approach that trains a separate classifier for each different pair of labels, as this generally outperforms a one-vs-all approach, particularly in the case of similar classes. We experimented with linear, radial-basis and polynomial kernels, tuning each model and evaluating their performance.

Support vector machines are binary classifiers (with a response coded as y ∈ {−1, 1}), which attempt to find a vector of parameters α to minimize the regularized loss function



kernel function K(x, y).[10].

While SVMs were developed in the context of binary classification, they can be employed in many-class classification problems in two ways:

• One vs. one. In the ‘one vs. one’ (ovo) method, SVMs are trained. Each SVM serves to separate one pair of classes from each other. A query point is predicted to be in the class that is chosen most frequently, when the point is evaluated using all of the SVMs.[15]

• One vs. all. In the ‘one vs. all’ (ova) method, L SVMs are trained. Each SVM serves to separate one class from the rest of the classes. A query point’s predicted class is the one for which its signed distance to the decision boundary is greatest.[15]

MF created by gridsearchCV’ to find out the best hyperparameter for our data so that we are able to predict the classes of observations on our test-data. Which is shown in figure below :

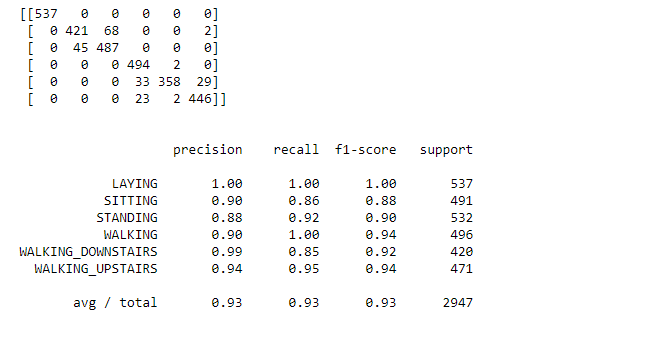
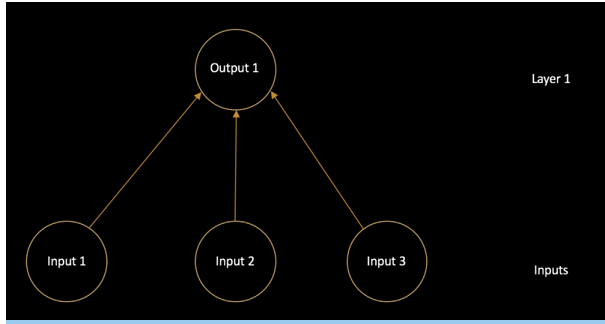


Fig: Prediction of classes of observations based on SVM algorithm

**3.2.3.3 CNN**

Neural Networks are a key piece of some of the most successful machine learning algorithms. The development of neural networks have been key to teaching computers to think and understand the world in the way that humans do. Essentially, a neural network emulates the human brain. Brains cells, or neurons, are connected via synapses. This is abstracted as a graph of nodes (neurons) connected by weighted edges (synapses)

The human brain consists of 100 billion cells called neurons, connected together by synapses. If sufficient synaptic inputs fire to a neuron, that neuron will also fire. We call this process “thinking”. We can model this process by creating a neural network on a computer. A neural network has input and output neurons, which are connected by weighted synapses. The weights affect how much of the forward propagation goes through the neural network. The weights can then be changed during the back propagation — this is the part where the neural network is now learning. This process of forward propagation and backward propagation is conducted iteratively on every piece of data in a training data set. The greater the size of the data set and the greater the variety of data set that there is, the more that the neural network will learn, and the better that the neural network will get at predicting outputs.



a neural network is a connected graph with input neurons, output neurons, and weighted edges. Let’s go into detail about some of these components:

**1) Neurons.** A neural network is a graph of neurons. A neuron has inputs and outputs. Similarly, a neural network has inputs and outputs. The inputs and outputs of a neural network are represented by input neurons and output neurons. Input neurons have no predecessor neurons, but do have an output. Similarly, an output neuron has no successor neuron, but does have inputs.

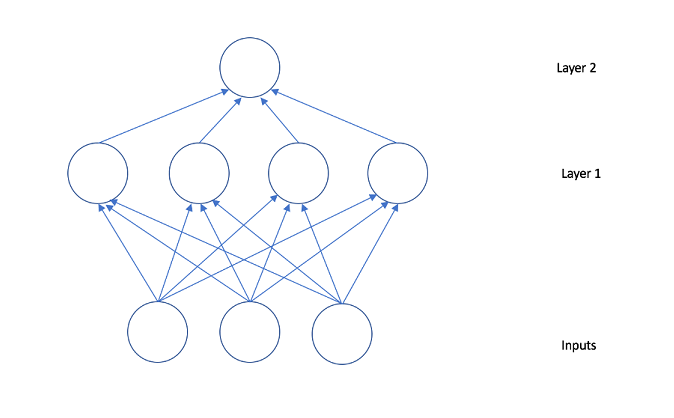
**2) Connections and Weights.** A neural network consists of connections, each connection transferring the output of a neuron to the input of another neuron. Each connection is assigned a weight.

**3) Propagation Function.** The propagation function computes the input of a neuron from the outputs of predecessor neurons. The propagation function is leveraged during the forward propagation stage of training.

**4) Learning Rule**. The learning rule is a function that modifies the weights of the connections. This serves to produce a favored output for a given input for the neural network. The learning rule is leveraged during the backward propagation stage of training.

**3.2.3.4 Deep Neural Networks**

A Deep Neural Network simply has more layers than smaller Neural Networks. A smaller Neural Network might have 1–3 layers of neurons. However, a Deep Neural Network (DNN) has more than a few layers of neurons. A DNN might have 20 or 1,000 layers of neurons.



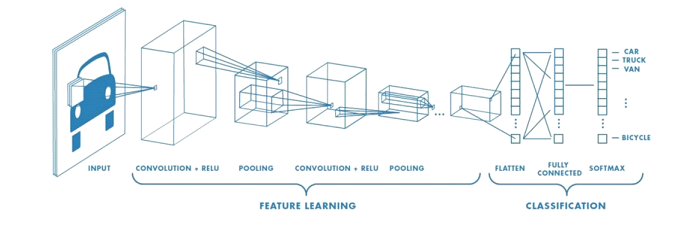
When designing features or algorithms for learning features, our goal is to separate the factors of variation that explain the observed data. These factors indicate separate influencing sources & are not combined by multiplication. Either they are unobserved objects/forces in the physical world that affect observable quantities or constructs in human mind providing simplified explanations or inferred causes of the observed data. They are concepts or abstractions that help us make sense of the rich variability in the data.

Deep learning solves this central problem in representation learning by introducing representations that are expressed in terms of other, simpler representations as it enables the computer to build complex concepts out of simpler concepts. Quintessential example of a deep learning model is the feed forward deep network, or multilayer perceptron (MLP). A multilayer perceptron is just a mathematical function formed by combining many simpler functions to map some set of input values to output values. Each application of a different mathematical function provides a new representation of the input.

Apart from learning right representation from data, another aspect is depth that enables computer to learn a multi-step program. Each layer of a representation is a state of the computer’s memory after simultaneously executing another set of instructions & that empowers networks with greater depth to execute more instructions in sequence. Later instructions can refer back to the results of prior instructions, so all the information in a layer’s activation don’t necessarily encode factors of variation that explain the input. Representation also stores state information that helps to execute a program that can make sense of the input and keep model processing organized.

Depth of a model can be viewed either based on number of sequential instructions (depth of Computational graph) OR based on correlation of concepts with each other (depth of Probabilistic modeling graph). Neither there is a single correct value for the depth of an architecture, nor is there a consensus about how much depth a model requires to qualify as ‘deep’. However, Deep Learning can be safely regarded as the study of models that involve a greater amount of composition of either learned functions or learned concepts than traditional machine learning does.

**3.2.3.5 Convolutional Neural Networks:**



In machine learning, a convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural networks.

CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing. They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.

Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage. artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics.

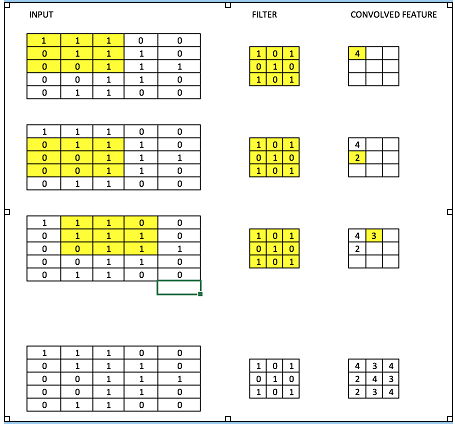
Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field.

CNNs use relatively little pre-processing compared to other image classification algorithm. This means that the network learns the filters that in traditional algorithms were hand-engineered. This independence from prior knowledge and human effort in feature design is a major advantage.

**Terms used in convolution neural networks:**

**1)Filter**

A filter in a CNN is like a weight matrix with which we multiply a part of the input image to generate a convoluted output. Let’s assume we have an image of size 28\*28. We randomly assign a filter of size 3\*3, which is then multiplied with different 3\*3 sections of the image to form what is known as a convoluted output. The filter size is generally smaller than the original image size. The filter values are updated like weight values during backpropagation for cost minimization. Consider the below image. Here filter is a 3\*3 matrix which is multiplied with each 3\*3 matrix section of the image to form the convolved feature.



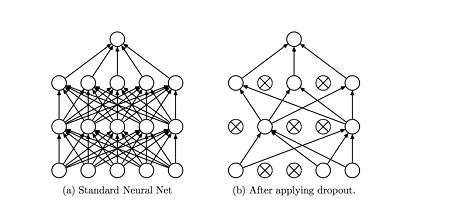
**2) Learning Rate**

The learning rate is defined as the amount of minimization in the cost function in each iteration. In simple terms, the rate at which we descend towards the minima of the cost function is the learning rate. We should choose the learning rate very carefully since it should neither be very large that the optimal solution is missed and nor should be very low that it takes forever for the network to converge.

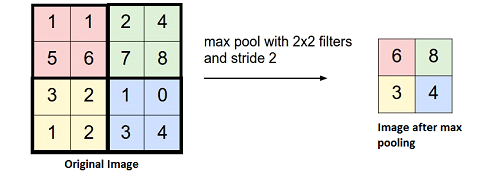
**3) Batches** – While training a neural network, instead of sending the entire input in one go, we divide in input into several chunks of equal size randomly. Training the data on batches makes the model more generalized as compared to the model built when the entire data set is fed to the network in one go.

**4) Epochs** – An epoch is defined as a single training iteration of all batches in both forward and back propagation. This means 1 epoch is a single forward and backward pass of the entire input data. The number of epochs you would use to train your network can be chosen by you. It’s highly likely that more number of epochs would show higher accuracy of the network, however, it would also take longer for the network to converge. Also you must take care that if the number of epochs are too high, the network might be over-fit

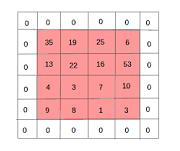
**5) Dropout** – Dropout is a regularization technique which prevents over-fitting of the network. As the name suggests, during training a certain number of neurons in the hidden layer is randomly dropped. This means that the training happens on several architectures of the neural network on different combinations of the neurons. You can think of drop out as an ensemble technique, where the output of multiple networks is then used to produce the final output.



**6) Pooling** – It is common to periodically introduce pooling layers in between the convolution layers. This is basically done to reduce a number of parameters and prevent over-fitting. The most common type of pooling is a pooling layer of filter size(2,2) using the MAX operation. What it would do is, it would take the maximum of each 4\*4 matrix of the original image



**7)Padding** – Padding refers to adding extra layer of zeros across the images so that the output image has the same size as the input. This is known as same padding.



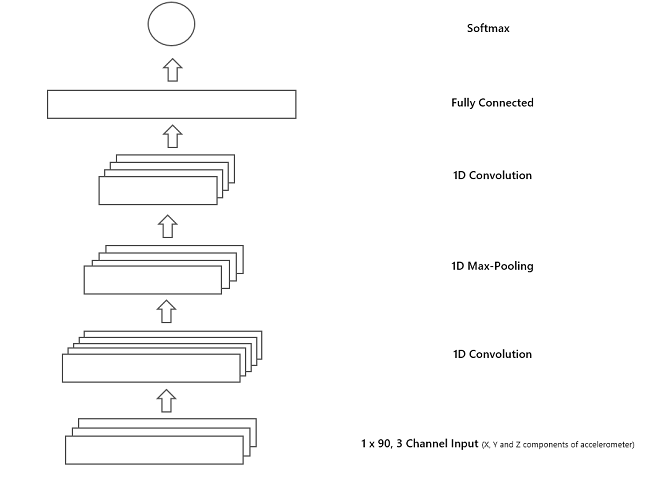
After the application of filters, The convolved layer in the case of same padding has the size equal to the actual image.

Valid padding refers to keeping the image as such an having all the pixels of the image which are actual or “valid”. In this case after the application of filters the size of the length and the width of the output keeps getting reduced at each convolutional layer.

Now we have to prepare the dataset in a format required by the CNN model. For doing this we define some helper functions to create fixed sized segments from the raw signal. The windows function will generate indexes as specified by the size parameter by moving over the signal by fixed step size. The window size used is 90, which equals to 4.5 seconds of data and as we are moving each time by 45 points the step size is equal to 2.25 seconds. The label (activity) for each segment will be selected by the most frequent class label presented in that window. The segment\_signal will generate fixed size segments and append each signal component along the third dimension so that the input dimension will be [total segments, input width and input channel. We will reshape the generated segments to have a height of 1 as we are going to perform one-dimensional convolution (depth wise) over the signal. Moreover, labels will be one hot encoded using get\_dummies function available in Pandas package. Now we have our data set in the desired format, and divided it into training and testing set (70/30) randomly.

**3.2.3.6 The CNN Model**

The figure below provides the CNN model architecture that we are going to implement using Keras having Theano as backend. The model will consist of one convolution layer followed by max pooling and another convolution layer. After that, the model will have fully connected layer which is connected to Softmax layer. The convolution and max-pool layers will be 1D or temporal.



The helper functions will be wrapper around Tensorflow functions to increase reuse and readability. The weight\_variable and bias\_variable will initialize Theano variables for our model layers. The apply\_depthwise\_conv (see Depthwise Convolution) will perform 1D convolution on each input channel separately and pass the output through ReLU activation function. Likewise, apply\_max\_pool will perform 1D max pooling on the output of convolution layer.

The first convolution layer has a filter size and depth of 60 (number of channels, we will get as output from convolution layer). The pooling layer’s filter size is set to 20 and with a stride of 2. Next, the convolution layer takes an input of max-pooling layer apply the filter of size 6 and will have a tenth of depth as of max-pooling layer. After that, the output is flattened out for the fully connected layer input. There are 1000 neurones in the fully connected layer as defined by the above configuration. The tanh function is used as non-linearity in this layer. Lastly, the Softmax layer is defined to output probabilities of the classThe negative log-likelihood cost function will be minimised using stochastic gradient descent optimizer, the code provided below initialize cost function and optimizer. It also defines the code for accuracy calculation of the prediction by model. Labels.

CNN model using a batch size of 10 for 5 training epochs. At each epoch, we will print out the model’s loss and accuracy on the training set. At the end of training, the model will classify the testing set instances and will print out achieved accuracy.

**CHAPTER FOUR**

**4.1 RESULTS & ERROR ANALYSIS**

In this section, we review the performance of each of the algorithms that we test and conduct error analysis on those that perform best

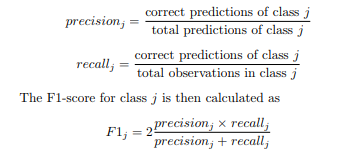
**4.1.1 Algorithm Comparison**

Our dataset contains roughly an equal number of observations for each of the six activities. Additionally, while specific applications of activity recognition may require that one or more activities be more accurately classified than others, given our general analysis we chose to weight each activity equally. As a result, we use the overall misclassification rate on the test data as our primary performance metric. The train and test errors for each of our analyses are displayed below:

|  |  |  |
| --- | --- | --- |
| Machine Learning Algrothim | Train Error | Test Error |
| kNN | 1.19696 | 18.0522565 |
| SVM | 0.78890097932 | 4.47913131999 |

Because we have a many-class problem with roughly uniformly distributed labels, the primary metric by which we assess our algorithms is their misclassification rate. Using the test data, we calculate the misclassification rate of each combination of feature engineering method and classificaton algorithm. These results, and the misclassification rate of each algorithm using the original (full, unmodified) data.

The F1-score is another useful evaluative metric. Because we are dealing with a many-class problem, there is a single precision, recall, and F1-score for every class. The precision and recall for class j are calculated as[16]



In Figure 4, we present the class-level F1-scores for each feature engineering method and classification algorithm.

FIG. 4. F1-scores by class for all feature engineering methods & classification algorithms

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Algrothim | WU | WD | Stand | Sit | Lay | Walking |
| kNN | 0.78 | 0.78 | ..81 | 0.77 | 0.98 | 0.78 |
| SVM | 0.94 | 0.92 | 0.90 | 0.88 | 1.00 | 0.94 |

Based on these scores, it seems that some classes are easier to predict than others. For example, the F1-scores for the ‘laying’ and ‘walking’ classes are usually higher than those for other classes, and the scores for the ‘sitting’ and ‘standing’ classes are generally lower. It seems that ‘laying’ and ‘walking’ have relatively distinct accelerometry and gyroscopic patterns, making them easier to predict. However, sitting and standing yield patterns that are similar to each other, making them harder to differentiate.

We also note that kNN performs significantly worse than the other methods we tried. We attribute this failure largely to the ‘curse of dimensionality’: the tendency for algorithm performance to degrade in high-dimensional space.[17] Because it is explicitly based on Euclidean distance between points, kNN is affected greatly, even at a modest number of dimensions.[16]

Although SoftMax regression did poorly with the original data set, obtaining a 10% error rate, its performance was markedly improved with the use of PC score vectors as features. With those features, it obtained a test error rate of about 5%. We surmise that the reduction in features (resulting in a reduction in over-fitting) is at the root of this significant drop in error

We are quite satisfied with the results produced by the two SVM-based methods. Using PC score vectors as features, both of those algorithms perform on-par with previous methods, obtaining test error rates of less than 4%. These results suggest that the data are close to linearly separable in the space of the first 200-300 principal components. We suspect that SVMs are especially effective because their regularized nature serves to combat variance.

The methods presented above do not reduce misclassification error beyond our benchmark. However, we achieve a 4% error rate with less than half the number of features that previous authors required to do the same. Moreover, the SVMs that employ the reduced set of engineered features perform roughly on-par (both in terms of misclassification rate and F1-score) with the SVMs that use the entire data set. Thus, by performing feature engineering/selection, we can both reduce the algorithm’s training time and decrease our model’s variance, without significantly increasing bias. In Figure 5, we present the number of features that minimized error on the validation data–and therefore the number we use to train our final models.

**4.1.1.1 Performance of linear kernel SVM**

Since the linear kernel SVM has a low misclassification rate and is computationally efficient to train, we decided to further diagnose its performance. For the purpose of feature selection, we applied PCA and experimented with training the model on a different number of principal components. The best result was obtained using the first 300 principal components, however this resulted in the same performance as simply applying linear kernel SVM to the original data. Since reducing the number of features did not improve the performance of the model, we chose to retain all 561 features. We then observed its training and test error, while varying the number of examples in the training data. The results are displayed in Figure below



Figure : Test vs. Train Errors

It is clear that the two lines are converging neither too close nor too far apart from each other as the number of training examples increases. This indicates that that there is no bias or variance issue with the model. Next, to examine its accuracy in classifying each activity, we computed the confusion matrix when trained and tested on the full train and test data:

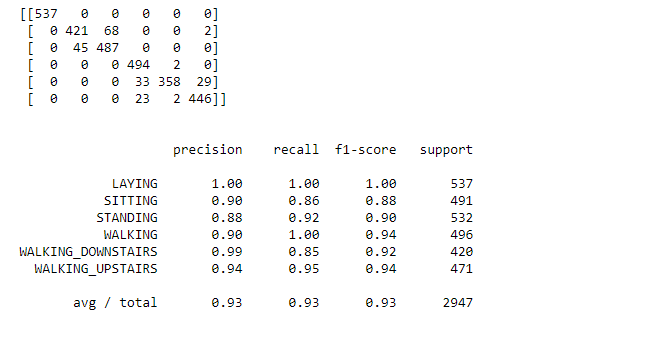


Figure : Confusion matrix for SVM with linear kernel

The activity misclassified most often is sitting, which has a misclassification rate of 11.6%, with almost all errors being incorrectly identified as standing. As expected, activities of motion are more likely to be mistaken with other activities of motion, and vice versa for static activities. In addition, after examining the specific observations for which sitting was misclassified, we observed that the errors mainly occurred during the transition from standing to sitting.

**4.2 IMPLEMENTATION OF THE PROJECT**

**4.2.1 ALGORITHMs**

**Model building using KNN/SVM:**

Step 1: Start

Step 2: Take training data from dataset

Step 3: Implement Classification algorithm; KNN and SVM Step 4: Build classification model

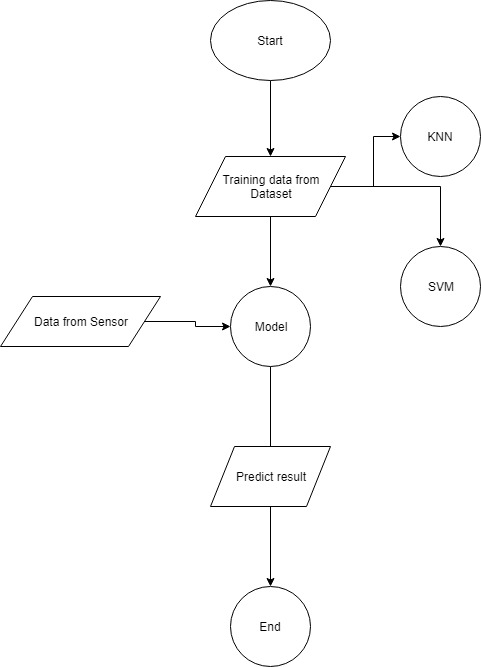
Step 5: Take sensor data

Step 6: Predict result

Step 7: End

**4.2.2 FLOWCHART**

**Model building using KNN/SVM**



**CHAPTER FIVE**

**EPILOGUE**

**5.1 RESULT**

Each and every step performed in this project has its own importance and values. What we learnt the most from this project is the practical implementations of neural network and integration of android app to self prepared modal. Although complete knowledge of one type of project cannot guarantee a total application to other areas, but it certainly broadens the area of knowledge. After completing the project, the following are the results obtained.

• A complete fully flegged dataset from real Time activity of users

• A working modal for 6 classes of activities.

• Addressed problems occur in detecting real time activity

• Result of extensive analysis of our algorithms on two datasets and also demonstrate how our algorithm can be used in mobile apps.

• Released full source code along model and android integration of apps.

**5.2 SCOPE**

This project has a wide range of applications in the field of human activity detection. Using this module and increasing it's class are it can be a great assist in the development field of human activity detection. This can also be a start or initiative to human activity and various releateable work like sensor based game playing, Human automation, Medical fields etc. The scope can be listed out as below:

 Sensor based Game development

Sensor Based patient movement check up

 Human Automation

 Traffic breakdown detection

In research to human activity detection

**5.3 PROBLEMS FACED AND SOLUTIONS**

**5.4 LIMITATIONS**

**5.5 CONCLUSIONS**

We demonstrate that it is possible to obtain error rates on par with previous HAR research, while using 60% fewer features. We obtain a 7% misclassification rate using gridSearchCV SVMs with linear kernels. This is the same rate obtained by previous authors, using one vs. all SVMs with Gaussian kernels.[28] We believe that we find success with only a relatively small number of PC-based features, because the data are close to linearly separable in the space of the first ∼200 PCs. Additionally, we believe that SVMs are an effective algorithm because their regularization penalty helps combat the primary issue we faced: high variance (due to a large feature space). Because the majority of the errors in our final model are due to confusion of the ‘sitting’ and ‘standing’ labels, we suggest that future authors should focus on creating a more effective classifier for only those classes. We also find that our algorithm likely still suffers from high variance; we believe that there are a number of avenues that future authors may pursue in order to remedy this, including regularization parameter tuning and even further feature reduction. The feature reduction that we achieve is a important improvement for a number of algorithmic reasons, including variance control and algorithm training time. It is particularly significant in the context of cell phones, because the devices have limited input/output and storage capabilities. Most significantly, if a fitness tracking application has lower data requirements, it will be less likely to cause problems with battery, storage, and data limits. As a result, the application will be more likely to be broadly adopted, as that is ultimately the goal.

**5.6 FURTHER ENHANCEMENTS**

The following enhancements are possible only if further time is allocated for the project.

 Building our own complete database for human activity detection.

 Sensor Based Game Development

 Extending classes for detection.

**5.7 REFRENCES**

[1] <https://www.quora.com/How-many-smartphones-users-are-in-the-world>

[2] <https://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones>

[3] S. H. E. Kim and D. Cook, Human Activity Recognition and Pattern Discovery, In IEEE Pervasive Computing (2009).

[4] L. Bao and S. S. Intille, Activity Recognition from User-Annotated Acceleration Data. PERVASIVE 2004, LNCS 3001.

[5] O. D. Lara and M. A. Labrador, IEEE Communications Surveys & Tutorials: A Survey on Human Activity Recognition using Wearable Sensors (Third Quarter 2013).

[6] O. D. Incel, M. Kose, and C. Ersoy, A review and taxonomy of activity recognition on mobile phones. In BioNanoScience 3, 145 (2013).

[7] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, Human Activity Recognition on Smartphones Using a Multiclass Hardware-Friendly Support Vector Machine. In Ambient Assisted Living and Home Care (2012).

[8] Bao, Ling and Stephen S. Intille. Activity recognition from user-annotated acceleration data, 2004.

[9] Mannini, A., Sabatini, A.M.: Machine learning methods for classifying human physical activity from on-body accelerometers. Sensors 10(2) (2010) 11541175

[10] T. H. G. James, D. Witten and R. Tibshirani, Introduction to Statistical Learning with Applications in R, 6th ed. (Springer, 2013).

[11] A. Ng and J. Duchi, CS229: Machine Learning - Course Notes. (Autumn 2016).

[12] <https://medium.com/@luckylwk/visualising-high-dimensional-datasets-using-pca-and-t-sne-in-python-8ef87e7915b>

[13] <http://jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf>

[14] <https://medium.com/@adi.bronshtein/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7>

[15] L. Mackey, STATS202: Data Mining and Analysis - Course Notes. (Autumn 2015).

[16] T. Hastie, R. Tibshirani, and J. Friedman, Elements of Statistical Learning, 3rd ed. (Springer, 2013).

[17] da Costa Cachucho, R. E. G., Moreira, J. M., and Gama, J. (2011). Activity recognition using, smartphone based, accelerometer sensors.

[18] Gyllensten, I. C. (2010). Physical activity recognition in daily life using a triaxial accelerometer. PhD thesis, KTH Royal Institute of Technology.

[19] Preece, S. J., Goulermas, J. Y., Kenney, L. P., Howard, D., Meijer, K., and Crompton, R. (2009). Activity identification using body-mounted sensors - a review of classification techniques. Physiological measurement.

[20] Bayat, A., Pomplun, M., and Tran, D. A. (2014). A study on human activity recognition using accelerometer data from smartphones. Procedia Computer Science, 34:450–457.

[21] Bao, L. and Intille, S. S. (2004). Activity recognition from user-annotated acceleration data. In Pervasive computing, pages 1–17. Springer.

[22] Parkka, J., Ermes, M., Korpipaa, P., Mantyjarvi, J., Peltola, J., and Korhonen, I. (2006). Activity classification using realistic data from wearable sensors. Information Technology in Biomedicine, IEEE Transactions on, 10(1):119–128.

[23] Subramanya, A., Raj, A., Bilmes, J. A., and Fox, D. (2012). Recognizing activities and spatial context using wearable sensors. arXiv preprint arXiv:1206.6869.

[24] Krishnan, N. C., Colbry, D., Juillard, C., and Panchanathan, S. (2008). Real time human activity recognition using tri-axial accelerometers. In Sensors, signals and information processing workshop.

[25] Lee, M. (2009). Physical activity recognition using a single tri-axis accelerometer. In Proceedings of the world congress on engineering and computer science, volume 1.

[26] Long, X., Yin, B., and Aarts, R. M. (2009). Singleaccelerometer-based daily physical activity classification. In Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE, pages 6107–6110. IEEE.

[27] Yang, J. (2009). Toward physical activity diary: motion recognition using simple acceleration features with mobile phones. In Proceedings of the 1st international workshop on Interactive multimedia for consumer electronics, pages 1–10. ACM.

[28] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, c ESANN 2013 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) (24-26 April 2013).