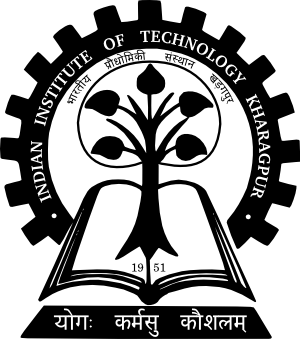
**Gesture Recognition : Wave Phone to Type**

****

The A-Team: **AbhayKhosla**

**AnshumanAgrawal**

**AnushreeBasu (*16AT91R17)***

**Arshdeep Singh Chahal**

**Jyotsna Singh**

**PrithviChandak**

**PriyaSingh**

**Sanket(13GG20022)**

*Submitted in partial fulfillment of the*

Machine Learning Project

*as a part of*

Machine Learning () - Autumn 2016

*under the supervision of*

Pabitra Mitra, Anirban Santara

November 16, 2016

**Gesture Recognition : Wave Phone to Type**

1. **Problem Definition:**

Recognizing gestures to identify digits drawn in the air while holding a cellular phone in hand. The data obtained from the accelerometer sensor of the phone has been used to identify digits using a pre-defined training set.

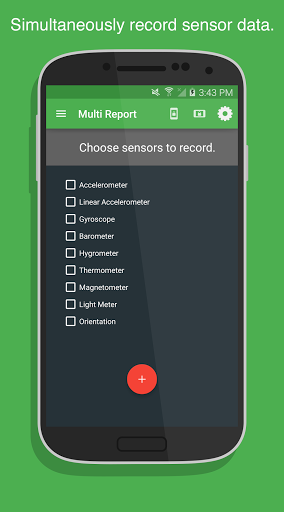
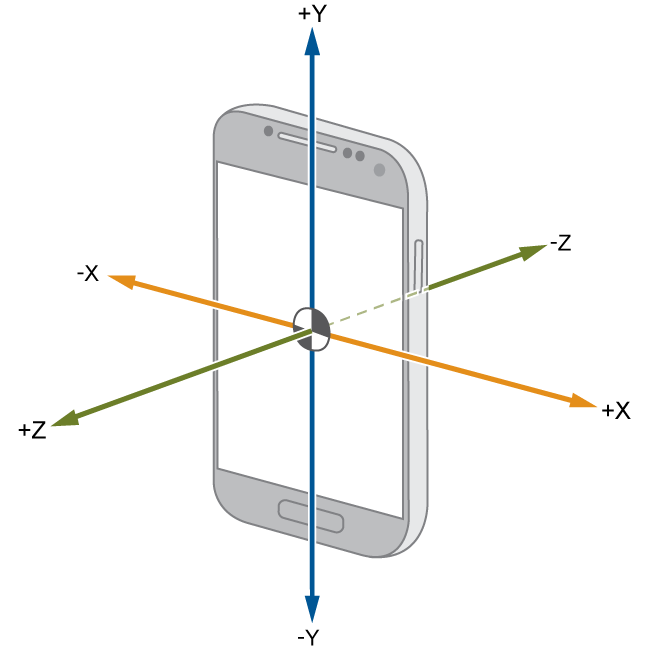
1. **Introduction:**

The touchscreen interface of smartphones and smartwatches limits the interaction with the phone or watch to the GUI-based point-and-click interactions. Similarly, current life-tracking software largely relies upon user input to collect data, which is often not effective. Software that collects and interprets data should not have the deficiency of needing manual input of data. By capturing acceleration data, it is possible to respond to gesture-based commands and to interpret daily activities as an automated data tracker.

The objective of this project is to interpret and respond to gestures using accelerometer data. The real-life implementation of this project would be to automatically detect gestures in a stream of accelerometer data and respond to user-defined gestures. The method has found a high amount of accuracy on the collected data when trained using various kinds of classifiers. However, we assume that in the real life implementation, the training set would grow with time with correct classification of new instances of the gesture, improving the performance over time.

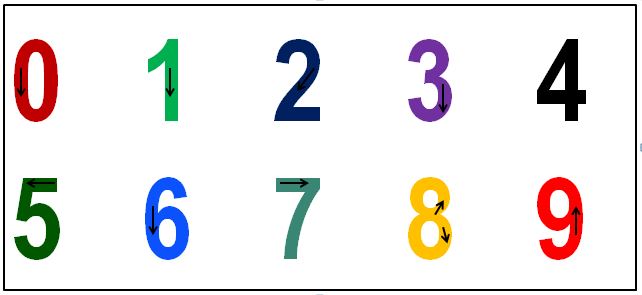
1. **Data and Preprocessing :**

To collect data, we used an Android Application (Physics Toolbox Suite) that records accelerometer data upon the click of an onscreen button and stores the data in a file with the gesture label provided by the user. We get a time series of data points along each axis – x, y and z for every gesture. Figure 1 shows the co-ordinates of the mobile device and the software used.



**Figure 1**: The co-ordinate data used from the mobile device, Physics Toolbox Suite used for capturing the data.

The time series of data points is then divided into ten equal sized windows for each axis. The mean and variance is calculated for each window and all values thus calculated are concatenated into a single feature vector which is of length 60. We obtained 300 examples for 10 different gestures – ‘0’,’1’,’2’,’3’,’4’,’5’,’6’,’7’,’8’,’9’. The direction convention used for capturing the data has been shown in the figure 2.



**Figure 2**: Number dataset with direction convention collected using a mobile device

1. **Classification Techniques:**

The collected data after pre-processing has been converted to a single feature vector which is finally used in various classifiers to determine the accuracy. The entire 3000 data has been divided into training and testing percentage of 70 % and 30 % respectively. The classifiers used have been detailed as follows:

* **SVM(5-fold cross validation mean score) :** In [machine learning](https://en.wikipedia.org/wiki/Machine_learning), support vector machines (SVMs, also support vector networks) are [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) models with associated learning [algorithms](https://en.wikipedia.org/wiki/Algorithm) that analyze data used for [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression analysis](https://en.wikipedia.org/wiki/Regression_analysis). Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-[probabilistic](https://en.wikipedia.org/wiki/Probabilistic_classification) [binary](https://en.wikipedia.org/wiki/Binary_classifier) [linear classifier](https://en.wikipedia.org/wiki/Linear_classifier). An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall on.

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the [kernel trick](https://en.wikipedia.org/wiki/Kernel_trick), implicitly mapping their inputs into high-dimensional feature spaces.

When data are not labeled, supervised learning is not possible, and an [unsupervised learning](https://en.wikipedia.org/wiki/Unsupervised_learning) approach is required, which attempts to find natural [clustering of the data](https://en.wikipedia.org/wiki/Data_clustering) to groups, and then map new data to these formed groups. The clustering algorithm which provides an improvement to the support vector machines is called support vector clustering[[2]](https://en.wikipedia.org/wiki/Support_vector_machine#cite_note-HavaSiegelmann-2) and is often used in industrial applications either when data are not labeled or when only some data are labeled as a preprocessing for a classification pass.

* **Decision Tree :** A decision tree is a [decision support](https://en.wikipedia.org/wiki/Decision_support_system) tool that uses a tree-like [graph](https://en.wikipedia.org/wiki/Diagram) or [model](https://en.wikipedia.org/wiki/Causal_model) of decisions and their possible consequences, including [chance](https://en.wikipedia.org/wiki/Probability) event outcomes, resource costs, and [utility](https://en.wikipedia.org/wiki/Utility). It is one way to display an [algorithm](https://en.wikipedia.org/wiki/Algorithm).

Decision trees are commonly used in [operations research](https://en.wikipedia.org/wiki/Operations_research), specifically in [decision analysis](https://en.wikipedia.org/wiki/Decision_analysis), to help identify a strategy most likely to reach a [goal](https://en.wikipedia.org/wiki/Goal), but are also a popular tool in [machine learning](https://en.wikipedia.org/wiki/Decision_tree_learning).

* **KNN (K used is three)** : In [pattern recognition](https://en.wikipedia.org/wiki/Pattern_recognition), the *k*-Nearest Neighbors algorithm (or *k*-NN for short) is a [non-parametric](https://en.wikipedia.org/wiki/Non-parametric_statistics) method used for [classification](https://en.wikipedia.org/wiki/Statistical_classification) and [regression](https://en.wikipedia.org/wiki/Regression_analysis). In both cases, the input consists of the *k* closest training examples in the [feature space](https://en.wikipedia.org/wiki/Feature_space). *k*-NN is a type of [instance-based learning](https://en.wikipedia.org/wiki/Instance-based_learning), or [lazy learning](https://en.wikipedia.org/wiki/Lazy_learning), where the function is only approximated locally and all computation is deferred until classification. The *k*-NN algorithm is among the simplest of all [machine learning](https://en.wikipedia.org/wiki/Machine_learning) algorithms.

The neighbors are taken from a set of objects for which the class (for *k*-NN classification) or the object property value (for *k*-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required.

Given a test example, the K-Nearest Neighbors algorithm outputs a classification based on which classes the majority of the “closest” K neighbor training examples belong to. This is essentially seeing which known gestures the new example looks like, and outputting that result

* **Neural Network(hidden layers =1 and single perceptron used) :** Neural Networks (also referred to as [connectionist](https://en.wikipedia.org/wiki/Connectionism) systems) are a computational approach which is based on a large collection of neural units loosely modeling the way the brain solves problems with large clusters of biological neurons connected by axons. Each neural unit is connected with many others, and links can be enforcing or inhibitory in their effect on the activation state of connected neural units. Each individual neural unit may have a summation function which combines the values of all its inputs together. There may be a threshold function or limiting function on each connection and on the unit itself such that it must surpass it before it can propagate to other neurons. These systems are self-learning and trained rather than explicitly programmed and excel in areas where the solution or feature detection is difficult to express in a traditional computer program.

Neural networks typically consist of multiple layers or a cube design, and the signal path traverses from front to back. Back propagation is where the forward stimulation is used to reset weights on the "front" neural units and this is sometimes done in combination with training where the correct result is known. More modern networks are a bit more free flowing in terms of stimulation and inhibition with connections interacting in a much more chaotic and complex fashion. Dynamic neural networks are the most advanced in that they dynamically can, based on rules, form new connections and even new neural units while disabling others.

The goal of the neural network is to solve problems in the same way that the human brain would, although several neural networks are much more abstract. Modern neural network projects typically work with a few thousand to a few million neural units and millions of connections, which is still several orders of magnitude less complex than the human brain and closer to the computing power of a worm.

* **Logistic Regression :** In [statistics](https://en.wikipedia.org/wiki/Statistics), logistic regression, or logit regression, or logit model is a [regression](https://en.wikipedia.org/wiki/Regression_analysis) model where the [dependent variable (DV)](https://en.wikipedia.org/wiki/Dependent_and_independent_variables) is [categorical](https://en.wikipedia.org/wiki/Categorical_variable). This article covers the case of [binary dependent variables](https://en.wikipedia.org/wiki/Binary_variable)—that is, where it can take only two values, such as pass/fail, win/lose, alive/dead or healthy/sick. Cases with more than two categories are referred to as [multinomial logistic regression](https://en.wikipedia.org/wiki/Multinomial_logistic_regression), or, if the multiple categories are [ordered](https://en.wikipedia.org/wiki/Level_of_measurement#Ordinal_type), as [ordinal logistic regression](https://en.wikipedia.org/wiki/Ordinal_logistic_regression).

Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a [logistic function](https://en.wikipedia.org/wiki/Logistic_function), which is the cumulative logistic distribution. Thus, it treats the same set of problems as [probit regression](https://en.wikipedia.org/wiki/Probit_regression) using similar techniques, with the latter using a cumulative normal distribution curve instead. Equivalently, in the latent variable interpretations of these two methods, the first assumes a standard [logistic distribution](https://en.wikipedia.org/wiki/Logistic_distribution) of errors and the second a standard [normal distribution](https://en.wikipedia.org/wiki/Normal_distribution) of errors.

Logistic regression can be seen as a special case of the [generalized linear model](https://en.wikipedia.org/wiki/Generalized_linear_model) and thus analogous to [linear regression](https://en.wikipedia.org/wiki/Linear_regression). The model of logistic regression, however, is based on quite different assumptions (about the relationship between dependent and independent variables) from those of linear regression. In particular the key differences of these two models can be seen in the following two features of logistic regression. First, the conditional distribution {\displaystyle y\mid x}y|X is a [Bernoulli distribution](https://en.wikipedia.org/wiki/Bernoulli_distribution) rather than a [Gaussian distribution](https://en.wikipedia.org/wiki/Gaussian_distribution), because the dependent variable is binary. Second, the predicted values are probabilities and are therefore restricted to (0,1) through the [logistic distribution function](https://en.wikipedia.org/wiki/Logistic_function) because logistic regression predicts the probability of particular outcomes.

* **Random Forest( no of estimators = 200) :** Random forests or random decision forests are an [ensemble learning](https://en.wikipedia.org/wiki/Ensemble_learning) method for [classification](https://en.wikipedia.org/wiki/Statistical_classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis) and other tasks, that operate by constructing a multitude of [decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning) at training time and outputting the class that is the [mode](https://en.wikipedia.org/wiki/Mode_(statistics)) of the classes (classification) or mean prediction (regression) of the individual trees. Random decision forests correct for decision trees' habit of [overfitting](https://en.wikipedia.org/wiki/Overfitting) to their training set.

1. **Results**

The above techniques were tested using 300 examples each of 10 different classes of gestures, ‘0’ , ‘1’ , ‘2’ , ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’. Out of the above techniques, the Naïve Bayes and KNN classified the gestures the best; they were able to achieve 99.25% accuracy. Other implementations, using Dynamic Time Warping, achieved 98% on testing examples. However, the gestures used are simpler (move left, move right) and perhaps more distinguishable than the ones we have used.

**Table 1**: Accuracy of classifiers

|  |  |
| --- | --- |
| **Classifier** | **Accuracy** |
| SVM(5-fold cross validation mean score) | 98.99% |
| SVM | 95.50% |
| Decision Tree | 91.76% |
| KNN | 99.25% |
| Neural Network | 97.75% |
| Logistic Regression | 98.5% |
| Random Forest | 98.87% |
| Naive Bayes | 99.25% |

1. **Conclusion and Discussion:**

The results obtained from classifying the gestures using different classifiers have shown a high rate of accuracy. The data collection has been done manually and it has been limited to 3000 data. With increased amount of data the fairness of accuracy can be better observed. The data collection has been limited to use of two mobile devices due to sensitivity issue and having uniformity among the data.

Thus, the future scope of this work includes testing the algorithm on a larger dataset and using other mobile devices with different accelerometer sensitivities. Along with this continuous gesture recognition through effective parsing of various gestures can be addressed. However, we can infer from the work done that gestures can be successfully recognized to type numbers automatically in the mobile device.

**References:**

[1] Klingmann, Marco. “Accelerometer-Based Gesture Recognition with the iPhone.”

[2] Prekopcsak, Zoltan (2008). “Accelerometer based real-time gesture recognition”. In Proceedings of the 12th International Student Conference on Electrical Engineering, Prague, Czech Republic.

[3] Jiayang Liu, Jehan Wickramasuriya, et al (2009). “uWave: Accelerometer-based Personalized Gesture Recognition and Its Applications.” In Pervasive Computing and Communications.