Classifying Human Activity using Sensor Data

Md Rasheduzzaman
Department of Computer Science
New Jersey Institute of Technology
mr468@njit.edu

***Abstract*—In this project I analyzed and experimented with data collected from mobile accelerometers sensor to classify human activity. First step of my project is to transform the high dimensional data into a lower dimension representation using principal component analysis (PCA) and linear discriminant analysis (LDA). Then four different classification algorithm k nearest neighbor (k-NN), Bayesian classifier with linear and quadratic decision boundary known as (LDA and QDA), and support vector machine (SVM) is used to classify the data. I also analyzed different parameter of these algorithm for best classification rate. Then compare their performance using confusion matrix and receiver operating characteristic (RO)C curve. After the performance comparison I found that QDA has accuracy of 0.67% over 4-fold data.**

***Index Terms*—PCA, LDA, QDA, k-NN, kernel SVM, and ROC.**

I. Introduction

The new generation mobile phone and wearable devices are equipped with state-of-the-art on-broad sensors like GPS sensor, gyroscope, digital compass, accelerometer etc. Data collected from these sensors has open up a new domain of research specifically medical data analysis. Sensor-based activity classification/recognition system design is one of them, which has potentially a lot health related application. By understating these sensor data we design better exercise program human or provide caretakers service to elderly by monitoring their activity or criminal activity.

Human activity recognition using mobile sensor data is an active research area, which has several benefits over other external or internal device. Now a days every body has a mobile phone and we almost carry it with us every where, which saves us the hustle of carrying any other devices. where specialized sensors can do only one thing, mobile has several other feature such as GPS, and Internet. Which can also provide emergency services when needed.

Motivated by some of the new wearable technology, in this paper I tried develop and compare several different machine learning system which can identity different human activity. Activity such as walking, sitting, standing, and . In this project I focus on offline classification of these activities and evaluated the performance of classification such as Bayesian classifier, k-nearest classifier, and SVM. Classification on mobile phone with large dataset is very costly because of limited memory, so I worked with small dataset which can be easily adapted on a mobile phone. Besides the analysis of the performance of the classifiers, I also experiment with five different subject we verify the classifier performance. Test result of all these experiment

shows that for time series data Bayesian classifier with quadratic decision boundary outperforms linear classifier, K-NN, and SVM.

The rest of the paper is organized as follows: In section 2 describe the dataset used for experiment. In section 3, I describe the dimension reduction and data normalization methods. In section 4 I describe all methods used in this project for experiment and their mathematical details. In section 5 I describe error metric used in these project for performance analysis. Last, but not least in section 6 I describe the experimental results.

II. Data description and Feature calculation

In this study we consider six activities: walking, jogging, ascending stairs, descending stairs, sitting, and standing. We selected these activities because many people in their daily routines perform them regularly. The activities also involve motions that often occur for substantial time periods, thus making them easier to recognize. In all cases data was collected from accelerometer in every 50ms and had 20 samples per second. The data collection was supervised by one of the WISDM [1] team members to ensure the quality of the data. The raw time series data is than transformed using different feature calculation methods. The features are described below, with the number of features generated for each feature-type noted in brackets:

- Average (3): Average acceleration (for each axis)
- Standard Deviation (3): Standard deviation (for each axis)
- Average Absolute Difference (3): Average absolute difference between the value of each of the 200 readings within the ED and the mean value over those 200 values (for each axis)
- Average Resultant Acceleration (1): Average of the square roots of the sum of the values of each axis squared $\sqrt{(x_i^2 + y_i^2 + z_i^2)}$ over the ED
- Time Between Peaks (3): Time in milliseconds between peaks in the sinusoidal waves associated with most activities (for each axis)
- Binned Distribution (30): We determine the range of values for each axis (maximum – minimum), divide this range into 10 equal sized bins, and then record what fraction of the 200 values fell within each of the bins.

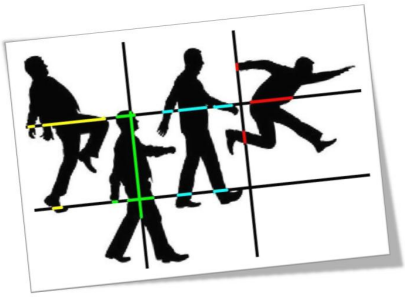

Figure 1: Human Activity

III. Feature normalization and dimensionality reduction

The sheer size of data presents a great challenge for machine learning and pattern classification algorithm. The main goal of dimensionality reduction algorithm is to finding the directions of maximum variance in high-dimensional data and project it onto a smaller dimensional subspace while retaining most of the information.. Both LDA and PCA are linear transformation methods. Principal Component Analysis, or simply PCA, is a statistical procedure concerned with elucidating the covariance structure of a set of variables. In particular it allows us to

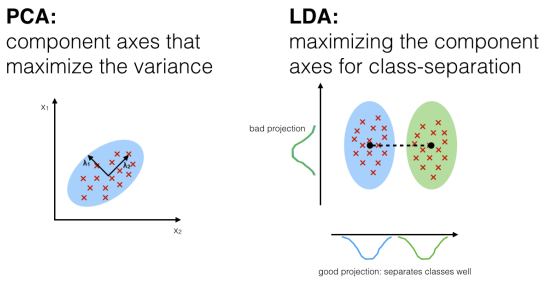

Figure 2: Dimensionality reduction

identify the principal directions in which the data varies. Whereas LDA also aims to find the directions that maximize the separation (or discrimination) between different classes, which can be useful in pattern classification problem (PCA "ignores" class labels). The standard PCA approach can be summarized as:

- Given $D = x_1, x_2, ..., X_n$.
- Compute $\mu = \frac{1}{n}\sum_{i=1}^{n} X^i$ and $\Sigma = \frac{1}{n}\sum_{i=1}^{n}(X^i - \mu)(X^i - \mu)^T$.
- Find k eigenvectors of $\sum$ with leargest eigenvalues: $\boldsymbol{\lambda}_1$, $\boldsymbol{\lambda}_2$, ..., $\boldsymbol{\lambda}_d$, called principal component.
- Construct the projection matrix: $z^i = ((X^i - \mu)^T\boldsymbol{\lambda}_1, ..., (X^i - \mu)^T\boldsymbol{\lambda}_k)$.

The standard LDA approach can be summarized in five simple steps:

- Compute the d-dimensional mean vectors for the different classes: $\mu_i = \frac{1}{n_i}\sum_{X\in\omega} X$.
- Compute the within-class scatter matrix: $S_w = \sum_{i=1}^{C} S_i, where S_i = \sum_{X\in\omega}(X-\mu_i)(X-\mu_i)^T$.
- Compute the between-class scatter matrix: $S_B = \sum_{i=1} Cn_i(\mu_i - \mu)(\mu_i - \mu)^T, where \mu = \frac{1}{n}\sum_{\forall X} X \sum_{X\in\omega}(X - \mu_i)(X - \mu_i)^T$
- Compute the eigenvectors ($\boldsymbol{e}_1$, $\boldsymbol{e}_2$, ..., $\boldsymbol{e}_d$) and corresponding eigenvalues ($\boldsymbol{\lambda}_1$, $\boldsymbol{\lambda}_2$, ..., $\boldsymbol{\lambda}_d$) for the scatter matrices.
- Sort the eigenvectors by decreasing eigenvalues and choose k eigenvectors with the largest eigenvalues to form a $k \times d$ dimensional matrix W (where every column represents an eigenvector).
- Use this $k \times d$ eigenvector matrix to transform the samples onto the new subspace. This can be summarized by the mathematical equation: $Y = X \times W$ (where X is a $n \times d$-dimensional matrix representing the n samples, and y are the transformed $n \times k$-dimensional samples in the new subspace).

Before we apply linear transformation methods we make sure that feature dimension are normalized.Feature standardization is process which makes sure that the features are re-scaled so that they'll have the properties of a standard normal distribution with $\mu = 0$ and $\sigma = 1$.

$$z = \frac{x - \mu}{\sigma}$$

IV. Classification methods

In classification problems, the task is to decide to which of a predefined, finite set of categories an object belongs. These classes or categories are known in advance; there is a finite number of them; and, in fact, there is usually only a small number of them. Each class or category has a name, which we refer to as the class label. We refer to the objects as instances. A classification system (or classifier) receives a description of the instance to be classified; decides to which class the instance belongs; and outputs the appropriate class label.

A. Bayesian classifiers

At each point x in feature space, choose class ω_i that maximizes (or minimizes) some measure $g_i(x)$. A useful way to represent classification is through

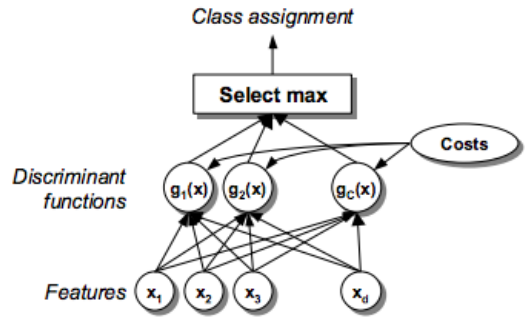

Figure 3: Multi-class Bayesian classifier

a set of discriminant function $g_i(x),\ i = 1, ..., C$, where a feature vector x is assigned to class ω_i if:

$$g_i(x) > g_j(x)\ \ \forall j \neq i$$

The three decision rules for Bayes classifier can be summarize as

- $g_i(x) = -\Re(\alpha_i|x)$ - general loss function
- $g_i(x) = P(\omega_i|x)$ - zero-one loss function
- $g_i(x) = P(x|\omega_i)$ - zero-one loss function

In order to simplify the calculation of $g_i(x)$, we can replace $g_i(x)$ with $f(g_i(x))$, where $f(\cdot)$ is monotonically increasing, which does not change the classification results.

$$\ln(g_i(x)) = \ln(P(x|\omega_i)) + \ln(p(\omega_i))$$

More common to use a single discriminant function instead of two:

$$g(x) = g_1(x) - g_2(x)$$

Decision rules divides the feature space in decision regions $R_1, ..., R_c$ separated by decision boundaries. Bayes classifier for normally distributed classes. The multivariate normal pdf is

$$p(x) = \frac{\exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu))}{(2\pi)^{\frac{n}{2}}\sqrt{|\Sigma|}}$$

For a given class denoted π_i, with prior probability $p(\pi_i)$:

$$p(\pi_i|x) = \frac{p(\pi_i)\exp(-\frac{1}{2}(x-\mu_i)^T\Sigma_i^{-1}(x-\mu_i))}{(2\pi)^{\frac{n}{2}}\sqrt{|\Sigma|}}$$

Assign pattern x to class π_i if $max_{1\leq j\leq g}\ g_i(x)$:

$$[-\frac{1}{2}\ln|\Sigma_j| - \frac{1}{2}(x-\mu_j)^T\Sigma_j^{-1}(x-\mu_j) + \ln(p(\pi_j))]$$

The Bayes classifier for Gaussian classes (general case) is quadratic and with equal covariance is linear.

B. K-Nearest Neighbors Algorithm

Nearest classification has a very simple basis: to classify a test item, find the item in the training set which is closest and assign the test item to the same class as the selected training item. K-nearest neighbor is very efficient at training time, since training simply consists of storing the training set. Testing is much slower, since it can potentially involve measuring the distance between the test point and every training point, which can make K-nearest neighbor impractical for large data sets. K-nearest neighbor algorithm is defined as:

- Given training data $D = \{\mathbf{x}_i, y_i\}$, distance function $d(\cdot,\cdot)$ and input $\mathbf{x}$
- Find $\{j_1, \ldots, j_K\}$ closest examples with respect to $d(\mathbf{x}, \cdot)$
  - (regression) if $y \in \mathbf{R}$, return average: $\frac{1}{K}\sum_{k=1}^{K} y_{j_k}$
  - (classification) if $y \in \pm 1$, return majority: $sign(\sum_{k=1}^{K} y_{j_k})$

Two of the shortcomings of the k-NN method is that all neighbors receive equal weight and the the number of neighbors must be chosen globally. We use a distance measure (e.g., Euclidean distance) to determine similarity. If we have a representation for which the distance measure is a reasonable measure of similarity, then the nearest neighbor method will

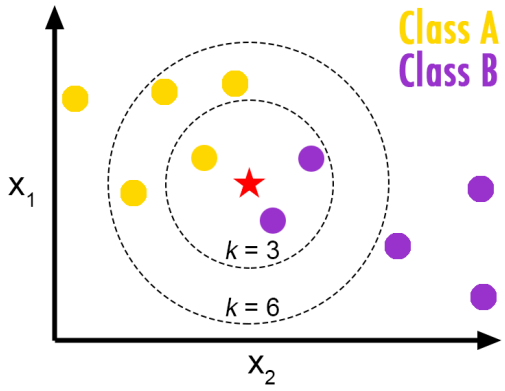

Figure 4: k nearest neighbour

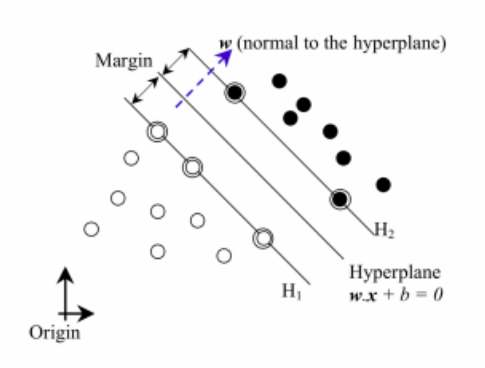

Figure 5: Linear Support Vector Machine

work well. There are variety of distance measuring function. Such as follows:

Euclidean distance function is the most common distance measure. defined as:

$$\sqrt{\sum_{i=1}^{N}(x_i - y_i)^2}$$

If an attribute is numeric, then the local distance function can be defined as the absolute difference of the values:

$$\sum_{i=1}^{N}|(x_i - y_i)|$$

this known as Manhattan distance.The easiest local distance function, known as the overlap function, returns 0 if the two values are equal and 1 otherwise:

$$dist(x, y) = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{otherwise} \end{cases}$$

this known as Hamming distance.

C. Support vector machine

The idea behind the SVM is to find a canonical hyper plane which maximize the margin that separates the given the classes of samples by minimizing the error. The same idea can be applied not just only for classification but also for regression problems. Given data with n variables and 1 target-variable $(x_1, y_1), ..., (x_n, y_n)$, where $x \in R^n$, $y \in R$. If the relation between x and y is approximately linear than, the model can be represented as:

$$f(x) = Wx + b$$

In order to make this a convex optimization problem feasible we introduce slack variable ξ to measure the deviation to training sample targets. We achieve this by formulation the problem as:

$$\mathbf{w}^*, b^*, \xi_i^* = \arg\min_{\mathbf{w},b,\xi_i} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m}\xi_i$$

subject to

$$\begin{aligned} \langle \mathbf{w}, \mathbf{x}_i \rangle + b; \geq +1 - \xi_i, \quad & y_i = +1, \\ \langle \mathbf{w}, \mathbf{x}_i \rangle + b; \leq -1 + \xi_i, \quad & y_i = -1, \\ \xi_i; \geq 0, \ \forall i. & \end{aligned}$$

For non-linear case the input patterns is transform into higher dimension feature space using kernel function, so that SVM can be performed:

$$f(x) = \sum_{i=1}^{N}\left(a_i, a_i^*\right) + b$$

$$f(x) = \sum_{i=1}^{N}\left(a_i, a_i^*\right).k\left(x_i, x\right)$$

where $k(.)$ is the kernel. Two common kernel is the radial basis function and polynomial kernel.

V. Error metric

The simplest form of evaluation is in terms of classification accuracy: the proportion of instances whose class the classifier can correctly predict. But how can we find this out? We take a dataset that contains instances whose class we already know. We ask the classifier to classify each of these instances in turn. Then we compare its prediction with the actual class of the instance. We take the proportion of correct classifications as an estimate of the accuracy of the classifier.

$$\textbf{Accuracy} = \frac{TP + TN}{FP + FN + TP + TN}$$

$$\textbf{Recall} = \frac{TP}{P}$$

$$\textbf{Precision} = \frac{TP}{FP + TP}$$

A. K-fold cross validation

Learning the parameters of a prediction function and testing it on the same data is a methodological mistake: a model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data. This situation is called over-fitting. To avoid it, it is common practice when performing a (supervised) machine learning experiment to hold out part of the available data as a test set. One such method is k-fold cross validation.

In k-fold cross-validation, the original dataset is first partitioned into subsets of equal size, . Each subset is used in turn as the test set, with the remaining subsets being the training set. In other words, first form the training set and is the test set; second form the training set and is the test set; and so on; finally, on the fold, form the training set and is the test set. The accuracy from each of the fold's are averaged to given an overall accuracy. A typical value for k is 10. This avoids the problem of overlapping test sets and makes very effective use of the available data.

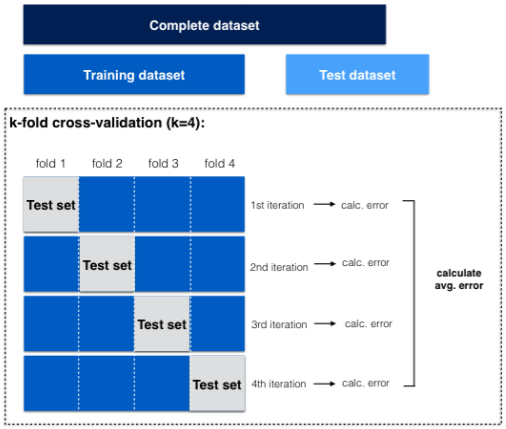

Figure 6: K-fold cross validation

VI. Experimental results

A. PCA and LDA

First stage in the experiment was to reduce the dimension of the dataset. In order to choose appropriate number of dimension I applied the percentage of the variance accounted for by the i^{th} eigenvector as:

$$r_i = 100 * \frac{\lambda_i}{\sum_n^j \lambda_j}$$

I found that with 35 dimension that we covered almost 99% of data variance. Same number of dimension were kept for both PCA and LDA. During my analysis I found that for proposed experiment and dataset LDA performance better than PCA. Figure shows that reduced dimension viewed on 3-dimension. We can clearly see that LDA separates the class better than PCA. So, for rest of the experiment I worked with data reduced into lower

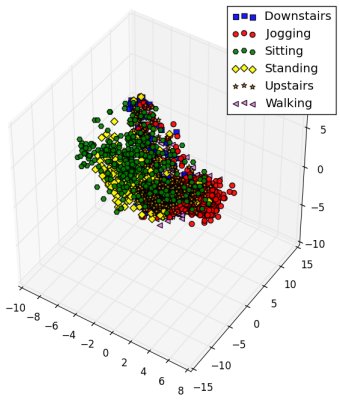

Figure 7: PCA in 3 dimension

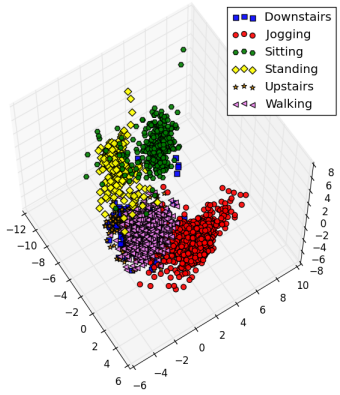

Figure 8: LDA in 3 dimension

dimension with LDA. Figure 7 and 8 shows the data in 3 dimension.

B. Bayesian classifier

Bayesian classifier is one of the most simplest classifier. It has two variation

- linear classifier, when covariance is equal
- quadratic classifier, general case

Here quadratic classifier means that the decision boundary has a quadratic shape. Table I and II show the result of the linear and quadratic classifier.

Table I: Confusion Matrix for LDA

		True label					
		Down	Jog	Sit	Stand	Up	Walk
Predicted label	Down	14	2	2	0	18	134
	Jog	5	455	0	0	0	15
	Stand	1	0	82	9	1	0
	Sit	1	0	2	67	7	0
	Up	9	20	0	1	39	113
	Walk	7	8	0	2	17	595

Table II: Confusion Matrix for QDA

		True label					
		Down	Jog	Sit	Stand	Up	Walk
Predicted label	Down	26	10	0	0	22	94
	Jog	7	476	0	0	4	8
	Stand	1	0	86	4	0	0
	Sit	9	0	1	54	2	0
	Up	18	16	1	1	41	117
	Walk	8	4	0	2	17	597

From the tables we see that quadratic classifier has slightly better classification rate than linear classifier. The average precision and recall score of LDA and QDA is 0.73, 0.76 and 0.74, 0.77. From both table we can wee that both LDA and QDA miss-classify class Down, Up and Walk.

C. k-nn

Most classifiers have parameters, and their values need to be chosen. The obvious example is the value of k for a $k - NN$ classifier. Its value can have a large effect on accuracy. In order to find best k value I experimented with 30 different k value. I found that with $k = 4$ we have best result of k-NN classifier. Figure 9 shows that result of accuracy achieved with different k value. For distance measure I used Euclidean distance.

Table III show the confusion matrix of k-NN classifier. Here again we see that k-NN miss-classify class Down, Up and Walk, but for other classes it

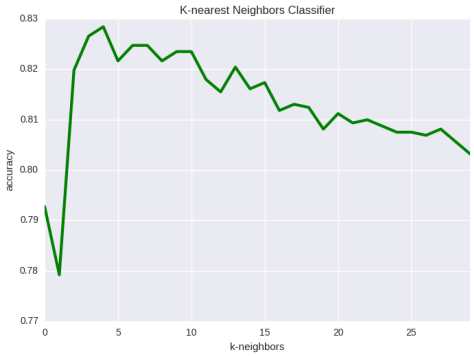

Figure 9: Accuracy rate at different k value

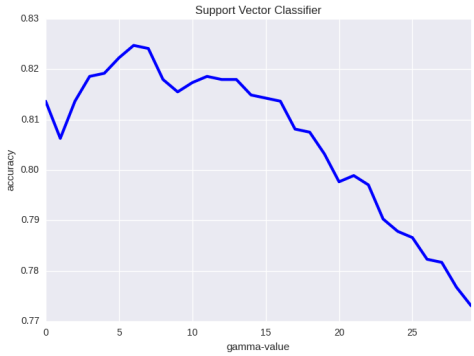

Figure 10: Accuracy rate at different gamma value

Table III: Confusion Matrix for k-NN

		True label					
		Down	Jog	Sit	Stand	Up	Walk
Predicted label	Down	59	6	2	0	29	50
	Jog	6	464	0	0	1	6
	Stand	0	0	79	5	0	0
	Sit	0	0	4	68	2	0
	Up	29	10	0	1	79	60
	Walk	36	1	0	1	42	586

Table IV: Confusion Matrix for SVM

		True label					
		Down	Jog	Sit	Stand	Up	Walk
Predicted label	Down	43	5	0	0	50	68
	Jog	1	478	0	0	5	5
	Stand	0	4	94	0	0	0
	Sit	2	1	1	71	1	1
	Up	11	17	0	0	73	89
	Walk	8	0	0	0	15	583

has over 93% accuracy. The average precision and recall score of k-NN is 0.81 and 0.82.

D. SVM

Just like K-NN, SVM also has one free parameter known as gamma value. Gamma value used in radial basis kernel in order to find similarity between samples. We need to choose this gamma value by experiment. From my experiment I found that when gamma value euqal to 0.6, SVM has the highest classification rate over all other classifier used in the project. Figure 10 shows the result of accuracy rate verses different gamma value.

Table IV show the confusion matrix of SVM classifier. Again we see that SVM miss-classify class Down, Up and Walk, but for other classes it has over 90% accuracy. The average precision and recall score of k-NN is 0.81 and 0.83.

E. k-fold cross validation

Complex model such as k-NN and SVM has tendency of over fitting the data set. In oder to find the best model for classification and solve the over fitting problem I applied the k-fold cross validation method. In my experiment I used 4-fold validation, because of small number of samples. After applying the 4-fold (figure 6) cross validation I found that both k-NN and SVM generalized the data more than the LDA and QDA and also QDA outperforms LDA in these settings. Using 4-fold cross validation i have also generated the precision and recall curves in order to support my arguments. Figure 11, 12, 13 and 14 shows the 4-fold precision and recall curves.

The are under the curve determines the accuracy of the model. from these four figures we see that QDA has the area 0.67 and LDA has area 0.66. Which tells us that QDA perofrme slightly better

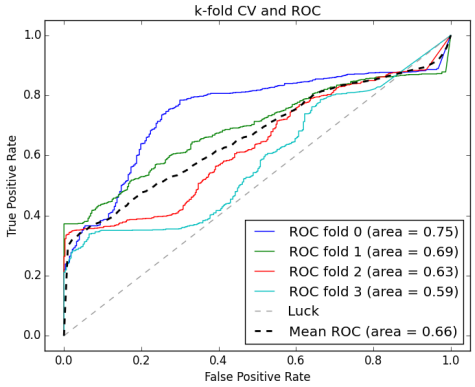

Figure 11: ROC curve for LDA

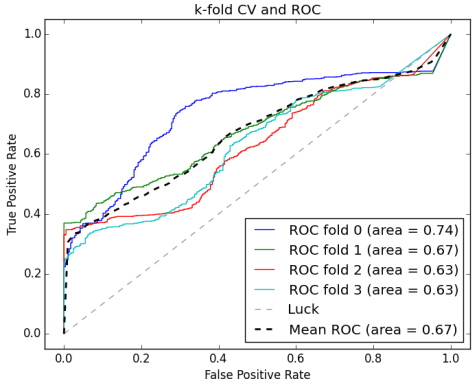

Figure 12: ROC curve for QDA

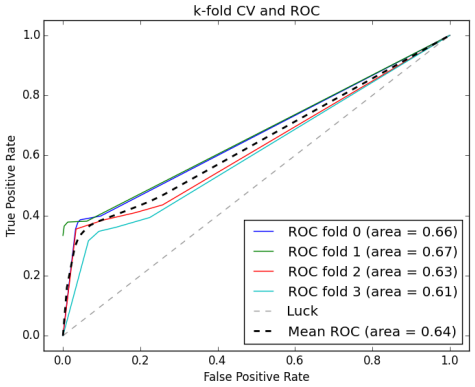

Figure 13: ROC curve for k-NN

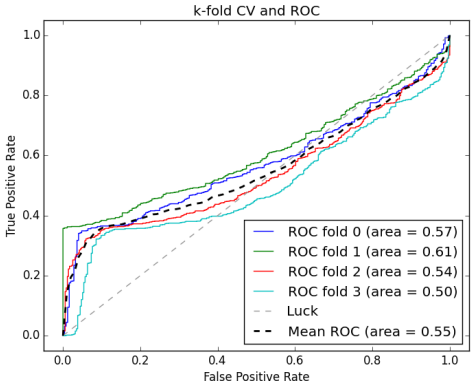

Figure 14: ROC curve for SVM

than LDA. k-NN has the area 0.64. We also that area for SVM si significantly lower than LDA, QDA, and k-NN. To be accurate it is 0.55. Which tell us that SVM over fitting the dataset. Which also proves the importance of k-fold cross validation for model selection and solving the over fitting problem.

VII. Conclusion

Through experiment I found that using only single 3-axis accelerometer we achieved fairly high accuracy. Out of four classifier we found that Bayesian classifier with quadratic decision boundary (Quadratic classifier) has the best classification rate. It outperform the complex model like SVM with very high margin. Activities such as walking plane surface or walking up or down stairs is very difficult to distinguish. Because all three of them have similar pattern. If we merge them and make them into single category called walking then the performance our four classifier on average achieved over 90% accuracy classifying the human activities.

An interesting extension of this project would be to see whether can we classify "short activities" (e.g. opening door or getting in the car) from accelerometer data. Another interesting extension can be building mobile applications based on accelerometer data to creating an intelligent calendar or identifying medical emergency.

REFERENCES

[1] Wisdm lab: http://www.cis.fordham.edu/wisdm/.

[2] Giuseppe Amato and Fabrizio Falchi. knn based image classification relying on local feature similarity. In *Proceedings of the Third International Conference on SImilarity Search and APplications*, SISAP '10, pages 101–108, New York, NY, USA, 2010. ACM.

[3] Ian Anderson, Julie Maitland, Scott Sherwood, Louise Barkhuus, Matthew Chalmers, Malcolm Hall, Barry Brown, and Henk Muller. Shakra: Tracking and sharing daily activity levels with unaugmented mobile phones. *Mob. Netw. Appl.*, 12(2-3):185–199, March 2007.

[4] Ling Bao and Stephen S. Intille. Activity recognition from user-annotated acceleration data. pages 1–17. Springer, 2004.

[5] Iyad Batal and Milos Hauskrecht. Boosting knn text classification accuracy by using supervised term weighting schemes. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 2041–2044, New York, NY, USA, 2009. ACM.

[6] Tomas Brezmes, Juan-Luis Gorricho, and Josep Cotrina. Activity recognition from accelerometer data on a mobile phone. In *Proceedings of the 10th International Work-Conference on Artificial Neural Networks: Part II: Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, IWANN '09, pages 796–799, Berlin, Heidelberg, 2009. Springer-Verlag.

[7] F.F. Chamasemani and Y.P. Singh. Multi-class support vector machine (svm) classifiers – an application in hypothyroid detection and classification. In *Bio-Inspired Computing: Theories and Applications (BIC-TA), 2011 Sixth International Conference on*, pages 351–356, Sept 2011.

[8] Yongwon Cho, Yunyoung Nam, Yoo-Joo Choi, and We-Duke Cho. Smartbuckle: Human activity recognition using a 3-axis accelerometer and a wearable camera. In *Proceedings of the 2Nd International Workshop on Systems and Networking Support for Health Care and Assisted Living Environments*, HealthNet '08, pages 7:1–7:3, New York, NY, USA, 2008. ACM.

[9] Narayanan C. Krishnan and Diane J. Cook. Activity recognition on streaming sensor data. *Pervasive Mob. Comput.*, 10:138–154, February 2014.

[10] Narayanan C. Krishnan, Colin Juillard, Dirk Colbry, and Sethuraman Panchanathan. Recognition of hand movements using wearable accelerometers. *J. Ambient Intell. Smart Environ.*, 1(2):143–155, April 2009.

[11] Jennifer Kwapisz, Gary Weiss, and Samuel Moore. Activity recognition using cell phone accelerometers. In *Proceedings of the Fourth International Workshop on Knowledge Discovery from Sensor Data, Washington DC.*, 2010.

[12] Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L. Littman. Activity recognition from accelerometer data. In *Proceedings of the 17th Conference on Innovative Applications of Artificial Intelligence - Volume 3*, IAAI'05, pages 1541–1546. AAAI Press, 2005.

[13] P. Soucy and G.W. Mineau. A simple knn algorithm for text categorization. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 647–648, 2001.

[14] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[15] Jun Yang. 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*, IMCE '09, pages 1–10, New York, NY, USA, 2009. ACM.

[16] Hao Zhang, A.C. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2126–2136, 2006.