Sensor Based Human Activity Recognition using Multi-Label Learning

BTP Semester Report

Rahul Kumar, Imroj Qamar and Jaskaran Singh Virdi
CSE 4th Year, IIT Ropar
Email : {krahul, imrojq, sjaskaran} @iitrpr.ac.in

Abstract—**The automatic recognition of human actions and goals from a series of sensor data based observations of their actions and environmental conditions has great applications in field of healthcare like early detection of diseases, developing assistive technology, real world like games and other Smart Systems using Machine Learning and Data Mining Techniques. This project aims to develop models for Sensor Based Human Activity Recognition (S-HAR) using Multi-label Learning. In this report, we study and analyze the performance of some of the popular Machine Learning approaches in Multi-label Learning on 2 standard Activity Recognition datasets and 1 bioinformatics dataset. We also present a promising direction for future in this project.**

Index Terms—*(Human Activity Recognition (HAR), Multi-Label Learning)*

1. Introduction

Sensor Based Human Activity Recognition aims to develop a generic model for recognition of human activities based on sensor data collected from Smart Homes. These Smart Homes are real houses with sensors to monitor the activities of its residents. The sensors used are ambient sensors most of which are binary and get triggered when a corresponding event is detected in their vicinity without any invasion of the privacy of the residents. Some of the most commonly used sensors include Pressure Mats, Proximity Sensors, Contact Sensors, Temperature Sensors, Motion Sensors and Infrared Sensors etc. Video Cameras are not used for privacy concerns. These Smart Homes help generate sensor data, which are more similar to the data collected in real world than the sensor data collected from laboratory settings.

There are 3 phases in an Activity Recognition Framework. The first phase involves collection of raw sensor data from smart homes from various sensors. The second phase involves processing the raw sensor data, extracting useful features and annotating the data with true activities for multi-label learning. The third phase involves developing Activity Recognition models based on supervised learning techniques using the labeled data as input. Typically, one deals with multi-class classification data. However, in Activity Recognition we generally deal with multi-label classification. The difference between multi-label classification and multiclass classification is that the former is a more generalized form of the latter. In multi-label classification, for each instance multiple target labels can be assigned whereas in multiclass classification each instance can only be assigned a single target label. In the case of human activity recognition, multi-label classification models assign a subset of target labels (activities) from the complete set of activities to an input instance.

This project focuses on the third phase in the activity recognition framework and aims to develop a supervised Multi label learning technique for Activity Recognition. Based on a sensory input, a smart home resident can be doing concurrent activities or multiple residents can be doing different activities that necessitate the need for multi-label learning in Activity Recognition. The two standard Center for Advanced Studies in Adaptive Systems (CASAS)[1] datasets collected from two Smart Homes over a period of two months are taken as input. We would also like to experiment with the Activity Recognition with Ambient Sensing (ARAS) [2] datasets.

In the report, in Section 2, we review the datasets used for the experiments, Section 3 discusses about the evaluation metrics used for measuring the performance of the multi-label learning models, Section 4 reviews the basic literature in multi-label learning and discusses some popular existing approaches, Section 5 presents the performance of the multi-label learning approaches discussed in the literature on the datasets and concludes the report. Section 6 presents a promising direction for future work on this project.

2. Datasets

In this paper for the purpose of comparison between algorithms, three datasets, one from bioinformatics data and other two are from activity recognition data are used. Details of the datasets are given as follows:

a) Yeast Gene Functional Analysis: It is one of the most well studied dataset for multi-label learning in the literature. Algorithms are evaluated by predicting the gene function class of the Yeast. Each gene can be part of the more than one of the 14 main functional classes. Each gene is represented by a 103 dimensional feature vector.

b) Cairo and t2 dataset: These are smart home activity recognition datasets where the data is collected using ambient sensors and labelled manually. Each feature vector contains the normalized value of the sensors response and their corresponding activity labels.

The specific properties for the datasets have been shown in Table 1.

TABLE 1: Data Set Statistics

Data Set	***Size Data Set***	***Features***	***Labels***	***Cardinality***	***Density***
Yeast	2417	103	14	4.2371±1.5708	0.2648±0.0982
Cairo	600	45	13	1.0667±0.2497	0.0821±0.0192
t2	12637	50	16	1.2457±0.5728	0.0779±0.0358

3. Evaluation Metric

Evaluation metrics used in multi-label learning in general are different from those used in single label learning problem, popular evaluation metrics for multi-label learning includes hamming loss, subset loss, ranking loss, coverage and F-measure. However in our paper we will not be using the metrics which use relative ranking between labels as the labels in the Cairo and t2 dataset are not ranked. Let N denote number of instances, K denote the number of labels and h denote the multi label classifier. For an instance x_i and label set Y_i the evaluation metrics used can be described as:

1. *Hamming Loss:* This represents the average of the number of wrongly predicted labels for a given instance. For a given dataset this value is averaged over all instances. Lower the value of hamming value, better is the performance. Mathematically it can be expressed as:

$$hLoss(h) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{K}|h(x_i)\Delta Y_i|$$

2. *Subset Loss:* This represents number of time the set of labels for an instance is predicted wrong. Subset loss is a stricter metric for error calculation in the sense that it counts a predicted label set as one error even if only one of the label is wrong compared to hamming loss where average number of wrongly predicted label is counted. Mathematically it can be expressed as:

$$sLoss(h) = \frac{1}{N}\sum_{i=1}^{N} count(h(x_i) \neq Y_i)$$

3. *F Macro:* Harmonic Mean of precision and recall values calculated from the confusion matrix of the labels .Mathematically it can be expressed as :

$$\text{FMacro} = \frac{2 * precison * recall}{precision + recall}$$

4. *F Micro:* Average of the F score for each of the labels, where for each individual label the precision and recall is calculated by treating that particular as positive and all other labels as negative class. Mathematically it can be expressed as :

$$F - Micro = \frac{\sum_{i=1}^{K} F_i}{K}$$

and,

$$F_i = \frac{2 * precision_i * recall_i}{precision_i + recall_i}$$

4. Related Work

Currently existing techniques dealing with multi-label learning can be broadly classified into three categories.

A. Problem Transformation

In this approach, the given multi-label learning problem is tackled by transforming it into a well-established simpler learning scenario which is a set of binary classification problems, which can then be handled using single-class classifiers. A commonly used baseline algorithm of this type is Binary Relevance (BR)[3]. In BR, a binary classifier is trained independently for each label. The output of each individual classifier is combined to get the output for all labels on a test instance. It assumes that the labels are independent. This approach is fast and fairly simple but does not model label correlation and the label independence assumption is not correct in many cases. Another baseline approach which is used is the Label Powerset (LP)[4]. This method takes each unique combination of labels as a separate class and trains a single multi class classifier on these classes. This approach models label correlation but it is extremely slow due to the large number of labels and suffers due to the sparse nature and scarcity of the data.

1) Random k-Labelsets (RAKEL)[5] - This approach tries to combine the advantages of both BR and LP.

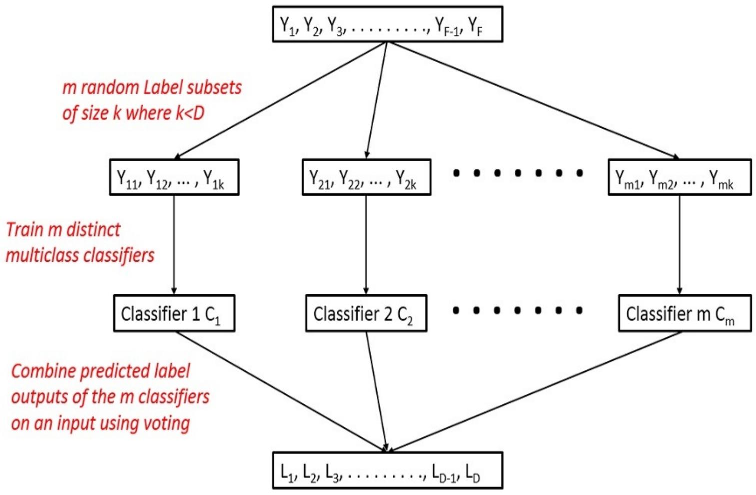
$Y_1, Y_2, Y_3, \ldots, Y_{F-1}, Y_F$
m random Label subsets of size k where k<D
$Y_{11}, Y_{12}, \ldots, Y_{1k}$
$Y_{21}, Y_{22}, \ldots, Y_{2k}$
$Y_{m1}, Y_{m2}, \ldots, Y_{mk}$
Train m distinct multiclass classifiers
Classifier 1 C_1
Classifier 2 C_2
Classifier m C_m
Combine predicted label outputs of the m classifiers on an input using voting
$L_1, L_2, L_3, \ldots, L_{D-1}, L_D$

Figure 1: Explanation for RAKEL algorithm.

Instead of performing an LP classification on the complete set of labels, LP is performed on subset of labels of size k. RAKEL iteratively constructs an ensemble of *m* Label Powerset (LP) classifiers each of which are trained on a small random subset of labels of size *k*. As shown in Figure 1, in each iteration, a distinct subset of labels, L_i of size *k* (called k-labelset) is selected from the complete set of labels and an LP classifier is trained from input to L_i. The total number of classes for the LP classifier is 2^k but classes which are not present in the training data can be ignored. On a new test input instance, RAKEL computes the output of all m classifiers and combines their predicted output using a voting mechanism. For each label, the ratio of the number of times the label is predicted true to the number of label subsets in which it is present among the selected *m* label subsets is calculated. Then each label for which the above calculated ratio is greater than certain threshold value *t* is classified as positive. The optimal *m*, *k* and *t* values are selected upon observing the experimental results. RAKEL achieves consistent lower Hamming Loss, Subset Loss and F 1 Micro for k=5. As m is increased, the results initially improve quickly after which there is very slow change. The Random selection of labels can impair the performance of the RAKEL algorithm as some labels might not be selected or some selected label sets might influence the results negatively. A future direction for improving the performance of this algorithm can be to develop a heuristic for selecting the labels instead of choosing them randomly to exploit the label correlation among the labels in a better manner. The experimental results obtained by varying the values of m and k on Cairo data set using 10 fold cross validation and SVM as the base classifier are shown below in Table 2. The values in cell are Hamming Loss, Subset Loss, F measure Macro, F measure Micro in order. The optimal values are used to get the evaluation measures on the Yeast and T2 datasets.

TABLE 2: Experimental Results on Cairo Dataset with *t* = 0.3

M ↓	**K →**		
	K = 3	***K=4***	***K= 5***
M = 2	0.0672±0.0030	0.0587±0.0042	**0.0574±0.0049**
	0.7350±0.0493	0.6150±0.0388	**0.5667±0.0737**
	0.5104±0.1155	0.5140±0.0635	**0.5237±0.0572**
	0.6934±0.0590	**0.7318±0.0609**	0.6831±0.0475
M = 3	0.0613±0.0044	0.0547±0.0033	**0.0486±0.0092**
	0.6533±0.0399	0.5417±0.0557	**0.4600±0.0876**
	0.5463±0.0986	0.5979±0.0623	**0.6317±0.0990**
	0.7121±0.0635	**0.7310±0.0324**	0.7176±0.0650
M = 4	0.0601±0.0105	0.0524±0.0061	**0.0479±0.0098**
	0.5850±0.0918	0.5033±0.0637	**0.4483±0.1090**
	0.5520±0.0855	0.6216±0.0558	**0.6261±0.0695**
	0.6579±0.0939	**0.7115±0.0443**	0.7058±0.0395
M = 5	0.0529±0.0103	0.0513±0.0088	**0.0469±0.0056**
	0.5100±0.0937	0.4867±0.1302	**0.4083±0.0387**
	0.5909±0.0808	0.6122±0.0636	**0.6456±0.0547**
	0.6911±0.0891	0.6862±0.0451	0.6708±0.0326
M = 6	0.0496±0.0063	0.0478±0.0074	**0.0447±0.0050**
	0.4583±0.0840	0.4233±0.0771	**0.3817±0.0616**
	0.6241±0.0633	0.6361±0.0508	**0.6766±0.0331**
	0.6878±0.0420	**0.6696±0.0449**	0.6771±0.0323
M = 7	0.0464±0.0090	0.0482±0.0111	**0.0460±0.0080**
	0.0430±0.0656	0.4233±0.1166	**0.4050±0.0653**
	0.6659±0.0741	0.6157±0.1010	0.6515±0.0638
	0.7055±0.0566	0.6799±0.0609	0.6804±0.0410
M = 8	0.0503±0.0093	0.0459±0.0070	**0.0455±0.0047**
	0.4667±0.1289	0.4167±0.0624	**0.3967±0.0680**
	0.6251±0.0766	0.6395±0.0822	**0.6586±0.0442**
	0.6745±0.0521	**0.6940±0.0377**	0.6663±0.0332
M = 9	0.0465±0.0068	0.0463±0.0037	**0.0436±0.0075**
	0.4217±0.0782	0.4267±0.0686	**0.3833±0.0745**
	0.6440±0.0509	0.6531±0.0405	**0.6748±0.0535**
	0.6959±0.0449	0.6803±0.0212	**0.6968±0.0378**
M = 10	0.0478±0.0082	0.0450±0.0052	**0.0433±0.0059**
	0.4283±0.1212	0.4033±0.0532	**0.3767±0.0378**
	0.6448±0.0549	0.6578±0.0365	**0.6860±0.0453**
	0.6716±0.0345	0.6950±0.0472	**0.6964±0.0536**

B. Algorithmic Adaptation

This is a type of approach in which an existing machine learning technique is modified to solve the problem of multi-label learning. Algorithms studied of these types are:

1) ML-KNN[6]: This lazy learning approach combines two of the famous machine learning techniques k-nearest neighbor (KNN) and maximum a posteriori (MAP) by using them sequentially. These two techniques are discussed in brief :

KNN: The principle behind k nearest neighbor is quite simple, instance are classified to any class based on their nearest neighbors. It is considered a lazy learning techniques because no generalization is made on the training data until a query is made on test data. It generally has two stages: first stage involving the determination of the k –nearest neighbor and second stage involving prediction using the class of the neighbors.

MAP: Maximum a posteriori uses the Bayes theorem to get a posterior distribution on the data. Bayes theorem can be expressed as:

$$posterior = \frac{prior * likelihood}{evidence}$$

This can be used in machine learning for classification as the posterior probability being the probability of belonging to a particular class y_i given an instance x_i and the instance will be classified as the class having maximum posterior probability.

$$p(y = y_j | x = x_i) = \frac{p(y = y_j) * p(x = x_i | y = y_j)}{p(x = x_i)}$$

ML KNN combines these two principles to obtain the probabilities of each of the labels in the label set. Before proceeding we define a set of notations for presenting ML-KNN mathematically. For a given instance x_i and its associated label set $Y_i \subseteq \mathcal{Y}$, where $\mathcal{Y}$ is the set of all possible labels. Let y_x be the label vector for x, where the l^{th} component $y_x(l)(l\in\mathcal{Y})$ be equal to 1 if $l\in Y$ and 0 otherwise and N(x) denote the set of k nearest neighbor of x from the training set as defined in. The membership counting vector can be defined as:

$$C_x = \sum_{a \in N(x)} y_a(l) \;,\; l \in \mathcal{Y}$$

At the time of prediction for each test set instance, first its k nearest neighbor in the training set are identified, using the neighbors C_t is calculated for the test set instance t. Following events are defined, H^l_1 is the event that t has label l, H^l_0 otherwise and E_j^l (j can be any value between 0 and k) is the event that exactly j of the neighbor has label l. Using these the resultant labels on y_t is defined as :

$$y_t(l) = \arg\max_{b \in \{0,1\}} P(H_b^l | E_{C_t}^l) \;,\; l \in \mathcal{Y}$$

And using the Bayesian rule it can be rewritten as:

$$y_t(l) = \arg\max_{b \in \{0,1\}} \frac{p\left(E_{C_t}^l | H_b^l\right) * p(H_b^l)}{p(E_{C_t}^l)}$$

Instead of using the Euclidean distance for k nearest neighbor other distance metric such as cosine distance can also be used. Similar to KNN k can be varied for the ML-KNN and optimal k can be selected.

Table 3 shows the 10 fold cross validation performance of ML-KNN on Cairo data for different values of k.

TABLE 3: Experimental Results on Cairo data ML-KNN

Data Set	***k=9***	***k=10***	***k=11***
Hamming Loss	0.0576±0.0226	**0.0567±0.0214**	0.0571±0.0230
Subset Loss	0.5650±0.1471	0.5567±0.1512	**0.5550±0.1453**
F-Micro	0.6762±0.2016	0.6798±0.1906	**0.6915±0.967**
F-Macro	0.6124±0.2154	**0.6149±0.2094**	0.6009±0.1967

It can be seen that the best value of hamming loss and F-Macro score is at k=10, however the best value of subset loss and F-Micro is at k=11.

2) Error-Correcting Code(ECOC) for Multi-label learning[7]: In this approach machine learning is treated as a communications problem. According to the paper by Dietterich and Bakiri[8], machine learning can also be treated analogous to a communications problem in which the identity of the correct output labels for a new example is being "transmitted" over a channel.

A typical communications problem has three components: Input Signal, the Communication Channel and the Output Signal.

Error-Correcting Codes are used to correct errors in the output signal at the receiving end. For this to happen, the input signal is first encoded using the error-correcting code and sent across the communication channel. This encoding is resilient to errors caused during transmission. The output signal is then decoded at the receiver's end.

When considering machine learning as a communications problem, the three components can be mapped as follows.

The Input Signal: The label corresponding to each data point is encoded using an error-correcting code. This encoded label space is the input signal.

The Communications Channel: The communications channel comprises of the learning algorithm, input features and examples. The errors are introduced in the labels by the finite training examples, poor choice of input features, and flaws in the learning process.

The Output Signal: The received label is decoded using Error-Correcting Code to its original label space. By encoding the label in an error-correcting code the system may be able to recover from errors depending upon the correction ability of the error-correcting code. We have used two error-correcting codes for the empirical study. They are:

a) Repetition Code (REP): Given a label *y* corresponding to the data point X_i, every bit in *y* is repeated $\left\lfloor \frac{M}{K} \right\rfloor$ times in *b*, which is the encoded codeword. Here, M is the length of the codeword *b* and K is the length of the codeword *y*.If M isn't a multiple of K, then (M mod K) bits are repeated one more time. Decoding takes a majority vote using received copies of each bit. This error-correcting code corrects up to $\frac{1}{2}\left\lfloor \frac{M}{K} \right\rfloor - 1$ bit errors in ***b***.

b) Hamming Repetition Code (HAMREP): Given a label *y* corresponding to the data point X_i , the label is encoded to a repetition code of a length which is a multiple of 4 followed by HAM(7,4) encoding scheme on this Repetition Code giving us the final encoded codeword *b*. The decoding takes place in the opposite way by first applying HAM(7,4) to get the repetition code and then using majority vote to get the original label.

Tables 4, 5 and 6 give the 10 fold cross validation results of the ECOC using SVM as the base classifier and Binary Relevance as the channel. In each case, the length(M) of the encoded codeword using REP encoding is three times the number of all possible labels(K) in the dataset and in case of HAMREP encoding it is seven times the number of all possible labels in the dataset.

TABLE 4:Experimental Results of ECOC on Yeast data(K=14)

Evaluation Measures	REP(M=42)	HAMREP(M=98)
Hamming Loss	0.18439±0.003681	0.18439±0.003681
Subset Loss	0.79811±0.024698	0.79811±0.024698
F-Micro	0.17443±0.001628	0.17443±0.001628
F-Macro	0.10274±0.003249	0.10274±0.003249

TABLE 5:Experimental Results of ECOC on Cairo data(K=13)

Evaluation Measures	REP(M=39)	HAMREP(M=91)
Hamming Loss	0.06678±0.004622	0.06678±0.004622
Subset Loss	0.80000±0.048426	0.80000±0.048426
F-Micro	0.84355±0.082641	0.84355±0.082641
F-Macro	0.73778±0.163669	0.73778±0.163669

TABLE 6:Experimental Results of ECOC on t2 data(K=16)

Evaluation Measures	REP(M=48)	HAMREP(M=112)
Hamming Loss	0.0286±0.000838	0.0286±0.000838
Subset Loss	0.30700±0.0082	0.30700±0.0082
F-Micro	0.68800±0.0085	0.68800±0.0085
F-Macro	0.56250±0.0217	0.56250±0.0217

From the above, we can conclude that the REP encoding technique and the HAMREP encoding technique have the same results and there is no effect on the three datasets. A possible reason could be that since the base classifier is a Gaussian SVM, which is a very strong classifier, the amount of error produced during the learning process is less than the correcting capability of the error-correcting codes which is two bits in case of REP and four bits in case of HAMREP. A possible solution could be using ECOC of larger length in the case of HAMREP. However, the process of encoding and decoding becomes expensive as the length of the encoded codeword increases. To avoid the increase in length of the codeword, one could change the base classifier to a slightly weaker one like a linear SVM to gain advantage from the error-correcting ability of the ECOC of small lengths. The REP approach shouldn't have any effect on any length (M) of encoding since its equivalent to binary relevance as only repetition is taking place.

C. Dependency Exploitation

This is a type of approach which exploits the inherent correlation/dependencies between instance features and labels to solve the problem of multi-label learning.
Algorithm studied of this type are:

1) Principal Label Space Transform[9]

In this approach the label subset is seen as a vertex of a k dimensional hypercube where k is the number of possible labels. In the hypercube view each label set Y is represented by a k dimensional vector $y \in \{0,1\}^k$ such that i^{th} component of y is one if $i \in Y$ otherwise zero. The hypercube view of label set has been shown in Figure 2. The number of possible vertices in a k dimensional hypercube is 2^k. However, the number of training examples are generally lower than this number so most of the vertices would be unoccupied, this giving rise to sparse hypercube. Now the multi-label algorithms do not need to learn for entire hypercube vertices instead they can focus on certain vertices and their neighborhood .Hypercube sparsity effectively allows dimensionality reduction in the label space without significant loss of performance.

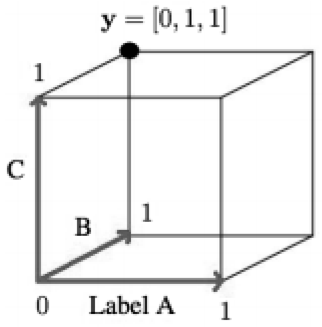

Figure 2: Hypercube view of label set.[9]

Any linear label space transform transformation process consists of three stages:
(i) Pre-processing: In this stage each point in the k dimensional hypercube is projected into a M-flat which is a M dimensional hyperplane described by a reference point o and orthonormal basis p_m where p_m^T are the rows of the M by k projection matrix P. Using these an instance $\{x_n,y_n\}$ is encoded to $\{x_n,h_n\}$ such that :

$$h_n = P(y_n - o)$$

(ii) Learning: Since after projection the points are no longer discrete the learning function would have to be regression instead of classification. The M-flat has M dimensions, hence learn function $r_m(x)$ from $\{(x_n,h_n[m])\}_{n=1}^{N}$ where m varies from 1 to M corresponding to each one of the M dimensions of the flat.
(iii) Prediction: After learning the regression function $r(x) = [r_1(x)\ r_2(x) \ldots\ldots r_M(x)]$ in the learning stage, for each test case we first compute r(x). Then transform it back to the hypercube space using the decoding function D where $D: R^M \rightarrow \{0,1\}^k$.
In linear label space transformation with round based decoding the decoding is done as $y = \text{round}(o + P^T r(x))$ where P and o are the projection matrix and reference point used in pre-processing stage respectively and round is a simple function which outputs 1 for number greater than 0.5 and 0 otherwise.
For linear label space transformation with round based decoding it can be proved that the upper bounded for the hamming loss between actual label set and predicted label set is sum of regression error and encoding error. Encoding error is the error caused by projecting to M-flat and re-projecting to hypercube and regression error is the error of the regression function learned.

$$\textit{hamming Loss} \leq \frac{4}{k}(\textit{encoding error} + \textit{regression error})$$

Principal Label Space transform aims to minimize the hamming loss by minimizing the encoding error. The objective function in PLST is the encoding error. Solving for the minimization of objective function gives the result o as the mean of all y values:

$$o = \frac{1}{N}\sum_{n=1}^{N} y_n$$

Using the value of o, Z is calculated as (y-o).Singular Value Decomposition of Z is used to calculate P.

$$Z = U\Sigma V^T$$

$$P = U_M^T$$

The largest M singular values are selected from Σ and their corresponding singular vectors from U are selected to get the transformation matrix. These values o and P are the values which gives the minimum encoding error.

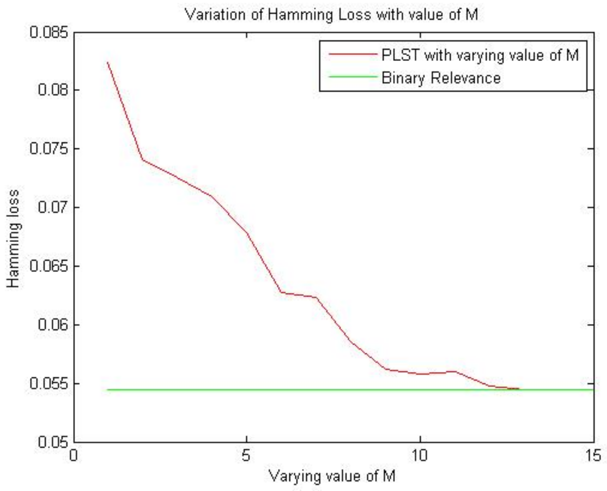

Figure 3 : Variation of Hamming Loss for values of M

Hamming Loss for PLST with ridge regression is calculated for different value and average of 10 fold cross validation is plotted in Figure 3 and compared to that of Binary Relevance. It can be seen that for low value of M the difference between PLST and Binary Relevance is high but as the value of M increases the difference decreases.

2) Conditional Principal Label Space Transform (CPLST)[10]

The above discussed feature unaware label space dimension reduction, PLST algorithm tries to minimize the encoding error, while a popular feature space dimension reduction technique called Canonical Correlation Analysis tries to minimize the learning error. The CPLST algorithm tries to achieve a combined objective and minimizes the Hamming Loss. It is a feature aware label space transformation technique which takes the feature space information i.e. key conditional correlations into account while making the transformation. The output labels Y are first mean shifted to Z. Then an orthogonal matrix transformation V is applied to transform the mean shifted space Z to transformed space ZV^T. Then m principal directions are selected to get *t* and a linear regression function *r* is learned from X to the transformed space *t*. On a test input first the hypothesis is predicted, then the decoding is done by applying the inverse transformation V^T, adding the mean shift, and then taking the round-off to get the decoded value. The transformation matrix V is obtained by taking the eigenvectors corresponding to largest eigenvalues of Z^THZ where H is $XX^\dagger$ such that $X^\dagger$ is the pseudo-inverse of X. This can be achieved by taking the SVD of Z^THZ. CPLST approach obtains small but consistent improvement over PLST for Hamming Loss as observed in the Cairo, T2 and Yeast datasets where the Hamming Loss for CPLST is lower compared to PLST. The regression function can be learnt for more complex relations between features and labels by applying the kernelization and regularization techniques.

5. Results And Discussion

The results of the four algorithms RAKEL with SVM classifier, ML-KNN, PLST with ridge regression and CPLST after applying 10-fold cross validation on each of the datasets Cairo, t2 and yeast have been tabulated in Table 7, Table 8 and Table 9 respectively.

TABLE 7: Experimental results on Cairo data

Evaluation Measures	RAKEL (m=7,k=5)	ML-KNN (k=10)	PLST (M=10)	CPLST (M=10)
Hamming Loss	**0.0460± 0.0080**	0.0567± 0.0046	0.0558± 0.0046	0.0554± 0.0038
Subset Loss	**0.4050± 0.0653**	0.5567± 0.1512	0.6283± 0.0509	0.6217± 0.0472
F1 Micro	0.6804± 0.0410	0.6798± 0.1906	0.7779± 0.0669	**0.7820± 0.0513**
F1 Macro	0.6515± 0.0638	0.6149± 0.2094	0.6720± 0.0728	**0.6730± 0.0583**

From Table 7, we can observe that on Cairo Dataset RAKEL performs better in terms of Hamming Loss and Subset Loss while F measures are better for CPLST.

TABLE 8: Experimental results on t2 data

Evaluation Measures	RAKEL (m=7,k=7)	ML-KNN (k=10)	PLST (M=10)	CPLST (M=10)
Hamming Loss	**0.0314± 0.0054**	0.0365± 0.0009	0.0416± 0.0012	0.0415± 0.0012
Subset Loss	**0.3186± 0.0559**	0.3729± 0.0107	0.4131± 0.0118	0.4123± 0.0114
F1 Micro	0.6112± 0.0301	**0.6836± 0.0085**	0.6740± 0.0099	0.6734± 0.0103
F1 Macro	0.4998± 0.0306	**0.5051± 0.0245**	0.3868± 0.0198	0.3920± 0.0236

From Table 8, we can observe that on t2 Dataset RAKEL still performs better in terms of Hamming Loss and Subset Loss however F measures for t2 Dataset are better for ML-KNN.

TABLE 9: Experimental results on yeast data

Evaluation Measures	RAKEL (m=7,k=5)	ML-KNN (k=10)	PLST (M=10)	CPLST (M=10)
Hamming Loss	0.2087± 0.0113	**0.1945± 0.0027**	0.2000± 0.0040	0.1998± 0.0042
Subset Loss	**0.8171± 0.0765**	0.8180± 0.0118	0.8482± 0.0184	0.8502± 0.0191
F1 Micro	0.1533± 0.0057	**0.1704± 0.0027**	0.1655± 0.0016	0.1657± 0.0015
F1 Macro	0.0976± 0.0029	**0.0998± 0.0033**	0.0914± 0.0027	0.0917± 0.0025

From Table 9, we can observe that on the Yeast dataset, ML-KNN obtains better performance measures in all evaluation metrics except Subset Loss.

We can’t reach a generic conclusion from the above results alone as results are different on different datasets. However from the above three dataset results, we can conclude that CPLST performs slightly better than PLST on all three datasets.

6. Future Directions

There are many ideas and approaches on which we wish to proceed for further research. Prominent one of those are listed here:

A. Rule Based Approach

In the case of smart home activity recognition it is easy to model the relation between sensors and activity in terms of rules. There are many rule based approaches in machine learning of which using decision trees[11] is quite a prominent approach. However, the problem with decision trees is that they are prone to overfitting. To overcome this problem an ensemble of trees can be used as described in the random forest approach. There is significant improvement in evaluation metrics using Random forest[12] for each label independently compared to all the approaches described in the paper. Instead of using Random forest independently for each label, we wish to find a modification in Random forest that can handle multi-label data.

TABLE 10: Evaluation metrics for Random forest on Cairo Data

Data Set	***Hamming Loss***	***Subset Loss***	***F1-Macro***	***F1-Micro***
Random Forest	0.025±0.005	0.232±0.006	0.891 ±0.025	0.883 ±0.030

It can be seen in Table 10 that the results obtained on Cairo data using Random forests are better than most of the other techniques results reported in the results section .

B. Feature Label Correlation

In the case of activity recognition it is highly possible that for a given activity certain sensors are more important than others which implies that in the data certain labels will be highly correlated with certain features. In Figure 4 we plot the Pearson correlation values for different features for different labels for Cairo dataset.

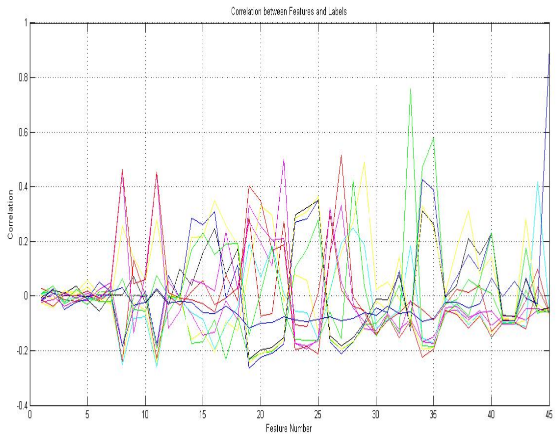

Figure 4: Correlation with features for different labels.

There are 45 features in Cairo dataset which are on x-axis and 13 labels whose correlation values with features are plotted with different colors. It can be seen in Figure 4 that for certain labels and features correlation is higher compared to others.

This can be used to give more importance to highly correlated features when predicting for a particular label. This was experimented on ML-KNN technique discussed above by multiplying the feature with the correlation value with the label while predicting for each label independently. Results are tabulated in Table 8.

TABLE 11: Comparison of ML-KNN using feature correlation based weights

Data Set	***Hamming Loss***	***Subset Loss***	***F1-Macro***	***F1-Micro***
Without Correlation	0.0576±0.022	0.5567±0.15	0.6798±0.19	0.6149±0.20
With Correlation	0.038±0.003	0.4167±0.02	0.736±0.06	0.7885±0.03

It can be seen that using feature correlation with ML-KNN significantly improves the evaluation metrics. Instead of multiplying the features directly with correlation value other modifications be thought of which encodes the correlation between features and labels.

7. Acknowledgment

We would like to use this opportunity to express our gratitude to our guide Narayanan C. Krishnan for his mentoring, supervision, guidance and support throughout the project.

8. References

[1] CASAS datasets-http://ailab.wsu.edu/casas/datasets.html

[2] Alemdar, Hande, Halil Ertan, Ozlem Durmaz Incel, and Cem Ersoy. "ARAS human activity datasets in multiple homes with multiple residents." In *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2013 7th International Conference on*, pp. 232-235. IEEE, 2013.

[3] Brinker, Klaus, Johannes Fürnkranz, and Eyke Hüllermeier. "A unified model for multi-label classification and ranking." In *Proceedings of the 2006 conference on ECAI 2006: 17th European Conference on Artificial Intelligence August 29--September 1, 2006, Riva del Garda, Italy*, pp. 489-493. IOS Press, 2006.

[4] Boutell, Matthew R., Jiebo Luo, Xipeng Shen, and Christopher M. Brown. "Learning multi-label scene classification." *Pattern recognition* 37, no. 9 (2004): 1757-1771.

[5] Tsoumakas, Grigorios, and Ioannis Vlahavas. "Random k-labelsets: An ensemble method for multi-label classification." In *Machine Learning: ECML 2007*, pp. 406-417. Springer Berlin Heidelberg, 2007.

[6] Zhang, Min-Ling, and Zhi-Hua Zhou. "ML-KNN: A lazy learning approach to multi-label learning." *Pattern recognition* 40, no. 7 (2007): 2038-2048.

[7] Ferng, Chun-Sung, and Hsuan-Tien Lin. "Multi-label Classification with Error-correcting Codes." In *ACML*, pp. 281-295. 2011.

[8] Dietterich, Bakiri. ”Solving Multiclass Learning Problems via Error-Correcting Output Codes”. *Journal of Artificial Intelligence Research 2* (1995) 263-286

[9] Tai, Farbound, and Hsuan-Tien Lin. "Multilabel classification with principal label space transformation." *Neural Computation* 24, no. 9 (2012): 2508-2542.

[10] Chen, Yao-Nan, and Hsuan-Tien Lin. "Feature-aware label space dimension reduction for multi-label classification." In *Advances in Neural Information Processing Systems*, pp. 1529-1537. 2012.

[11] Quinlan, J. Ross. "Induction of decision trees." *Machine learning* 1, no. 1 (1986): 81-106.

[12] Breiman, Leo. "Random forests." *Machine learning* 45, no.1(2001):5-32.