MACRO-CLASS SELECTION FOR HIERARCHICAL K-NN CLASSIFICATION OF INERTIAL SENSOR DATA

Corey McCall, Kishore Reddy and Mubarak Shah
Dept. of Electrical Engineering and Computer Science, University of Central Florida, Orlando, USA
shah@eecs.ucf.edu

Keywords: Macro-class Selection : Hierarchical Classification : Human Activity Recognition

Abstract: Quality classifiers can be difficult to implement on the limited resources of an embedded system, especially if the data contains many confusing classes. This can be overcome by using a hierarchical set of classifiers in which specialized feature sets are used at each node to distinguish within the macro-classes defined by the hierarchy. This method exploits the fact that similar classes according to one feature set may be dissimilar according to another, allowing normally confused classes to be grouped and handled separately. However, determining these macro-classes of similarity is not straightforward when the selected feature set has yet to be determined. In this paper, we present a new greedy forward selection algorithm to simultaneously determine good macro-classes and the features that best distinguish them. The algorithm is tested on two human activity recognition datasets: CMU-MMAC (29 classes), and a custom dataset collected from a commodity smartphone for this paper (9 classes). In both datasets, we employ statistical features obtained from on-body IMU sensors. Classification accuracy using the selected macro-classes was increased 69% and 12% respectively over our non-hierarchical baselines.

1 INTRODUCTION

Inertial Measurement Units (IMUs) have become pervasive in smartphones and consumer electronics devices, and can be employed to recognize human activities. In this paper, we attempt to classify a large number of confusing aerobic and cooking activities using statistical features computed from 9 degree-of-freedom IMUs. Most previous research in this area has focused on processing just a small number of either simple classes on the device itself such as in (Ganti et al., 2010) and (Saponas et al., 2008), or more complex classes on a dedicated server such as in (Iso and Yamazaki, 2008) and (Miluzzo et al., 2008). Although high classification accuracy is achieved, real-world applications would benefit from the ability to classify a large number of confusing classes using the minimal computational resources available on the device itself. This could allow for pervasive lifestyle monitoring of more complex scenarios such as exercise patterns, cooking habits, and disease symptoms, all of which have been shown to be recognizable using on-body IMUs in (Ermes et al., 2008), (Spriggs et al., 2009), and (Kim et al., 2009) respectively. In order to run these types of applications on commodity hardware, a low-cost classification method for a large number of confusing classes must be developed.

We examine a hierarchical version of the low-cost algorithm, k-Nearest Neighbor (k-NN). Because of its simplicity, traditional k-NN does not perform well when distinguishing between similar classes which tend to cluster together in feature space. This can be overcome by breaking the single k-NN classifier into a hierarchical set of simpler k-NN classifiers in which specialized feature sets are used at each node to distinguish within the mutually exclusive macro-classes defined by the hierarchy. For example, the two distinct actions *climbing stairs* and *descending stairs* in a dataset of aerobic actions may be easily distinguished from other classes such as *jumping* and *biking* when using a feature like mean forward acceleration. However, in the same feature space, these two actions are easily confused with one another. If mean forward acceleration is used to place these two actions in the same macro class, a better feature such as mean upward acceleration can be used at the second level to distinguish between the two actions.

This hierarchical classification process is illustrated in Figure 1, in which test sample X is classified into macro-class α, β, γ, or δ using a feature set, S_1, determined by feature selection. X is then classified among a smaller subset of classes using S_2, S_3, S_4, or

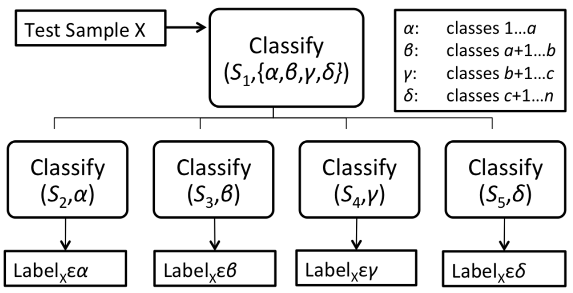

Figure 1: In this hierarchical model classifier, test sample X is classified into mutually exclusive macro-class α, β, γ, or δ using selected feature set S_1. X is then classified among a smaller set of classes using a reselected feature set S_i which better differentiates the classes within each macro-class.

S_5 (also determined by feature selection) depending on the macro-class. This two-tiered hierarchical design increases efficiency by dividing the training data and related k-NN time complexity by the four macro-classes, and increases classification performance by allowing normally confused classes to be grouped together and distinguished separately using a specialized set of features that differentiate them better than the original S_1. The keys to this method are: 1) ensuring maximum accuracy in the top level, 2) determining appropriate macro-classes, and 3) selecting quality feature sets.

The focus of this paper is the development of an algorithm that simultaneously determines good macro-classes and the features that best distinguish them, while attempting to maintain a high classification accuracy at the top level of the hierarchy. We modify a wrapper-model greedy forward selection algorithm, such that for each candidate feature set, k-means is used to cluster the mean-centers of each class according to the training set. A score is then determined by combination of the accuracy of the clustering, the number of clusters, and the evenness of the class distribution across the clusters. As single features are added at each iteration, the clustering accuracy increases while the score is maximized until an accuracy threshold is achieved. The accuracy of the training set is then considered adequate, and a modified scoring equation is used to optimize the class distribution while maintaining an accuracy above the threshold. Standard greedy forward selection is then used to select features for each classifier in the second level of the hierarchy.

The rest of this paper proceeds as follows. In Section 2, we discuss related work on macro-class selection. In Section 3, we discuss the macro-class selection algorithm. In Section 4, we discuss the datasets and features used to test the algorithm. In Section 5, we compare the non-hierarchical baseline results with those obtained with the hierarchical model. In Section 6, we conclude the paper with a discussion of our results.

2 RELATED WORK

The most relevant work in macro-class selection for hierarchical classification is given in (Wang and Casasent, 2008). An algorithm based on "weighted support vector k-means clustering" uses clustering to select macro-classes to form a hierarchical classifier for multi-class classification using a binary classifier at each node. This is similar to the one used in this paper in that similar classes are grouped together by clustering to build the hierarchy. However, we note two key differences. First, this method does not include feature selection, an integral part of our motivation to use multiple feature sets to improve classification performance of confusing classes. And second, this method builds a hierarchy with an undefined number of levels based on the number of classes (each node can handle exactly two classes/macro-classes), whereas our method is restricted to two levels. Our intuition is that we can minimize the misclassification of the testing data into the wrong macro-class by limiting the number of levels in the hierarchy to the minimum of two. Since we are attempting to improve the accuracy over a non-hierarchical baseline, any misclassification at the top level disqualifies the test sample from being correctly classified.

In the context of feature selection using unsupervised learning, (Zeng and Cheung, 2009) and (Law et al., 2004) use feature selection with mixture model clustering to successfully group unlabeled data. Unlike our greedy approach, the authors' focus on removing irrelevant or redundant features at each iteration of the algorithm. We choose a greedy algorithm because of its ability to find a smaller good solution by starting with an empty set, reducing the complexity of the resulting k-NN algorithm (perfect feature selection is considered intractable (Kohavi and John, 1997)). Additionally, we choose a wrapper-model algorithm, as opposed to filter-model, because of it's proven superiority in (Liu and Yu, 2005) and (Talavera, 2005). According to this research, the wrapper model gives better performance at the cost of a more computationally expensive algorithm, which is acceptable considering that the algorithm is only performed in the training phase.

We also review previous work done on the CMU-MMAC Dataset in (Spriggs et al., 2009) and (Fisher and Reddy, 2011). Although these papers do not use macro-classes, they provide quality baselines using more complex classifiers such as Support Vector Machines, Hidden Markov Models, and Neural Networks. We discuss these results in Section 6.

Overall, the research presented in this paper extends the small amount of previous work done in macro-class selection. The main contribution is an algorithm that simultaneously determines quality macro-classes and features for k-NN classification. The result is a set of macro-classes that can be used to build a hierarchical k-NN classifier that improves the overall accuracy of the model when there are a large number of confusing classes.

3 METHOD

We present our method in three parts. In Section 3.1, we present a straightforward feature selection algorithm that we use to select features for each classifier in the second level of the hierarchy after the macro-classes have been determined. In Section 3.2, we modify the algorithm for macro-class selection, which we use to select features for the classifier at the top level of the hierarchy, as well as the macro-classes that define the second level. In Section 3.3, we show how both algorithms are combined to build the final hierarchical classifier.

3.1 Base Feature Selection

Figure 2 shows a basic wrapper-model greedy forward selection algorithm for feature selection. Input to the algorithm are the training set $\mathbf{X}$, consisting of M examples, each with a pool of N scaled potential features, and $\mathbf{y}$, the corresponding label vector. The algorithm keeps track of accuracy A, and a selected feature set S. At each iteration, an exhaustive set of candidate feature sets is built by combining the current S with one of the potential features not in S. Each candidate set $S \cup \{i\}$ is then evaluated using a k-NN classifier with k-fold cross validation, where in this case, k is equal to 5% of the size of the training data. In this k-NN classifier, and all subsequently referenced in this paper, we use one nearest neighbors. If the maximum accuracy a achieved from testing each potential $S \cup \{i\}$ is less than A, S is returned as the selected feature set. Otherwise, A is updated to a, and the corresponding potential feature b is added to S.

In an attempt to further reduce the feature set and generalize it from the training data, we eliminate all features added to S after the final accuracy stopped increasing, as these features are assumed to overspecify the model to the training data. For example, if the

```
Input: $\mathbf{X} \in \mathbb{R}^{M \times N}$, $\mathbf{y} \in \mathbb{R}^{M}$
Output: $S$
 1: $[A,S] \leftarrow [0,\emptyset]$
 2: while $|S| < N$ do
 3:     $[a,b] \leftarrow [0,0]$
 4:     for all $i \in \{1,\ldots,N\} \setminus S$ do
 5:         $a_i \leftarrow \mathrm{KNN}(\mathbf{X}_{S\cup\{i\}},\mathbf{y})$
 6:         if $a_i > a$ then
 7:             $[a,b] \leftarrow [a_i,i]$
 8:         end if
 9:     end for
10:     if $a < A$ then
11:         break
12:     end if
13:     $[A,S] \leftarrow [a, S\cup\{b\}]$
14: end while
```

Figure 2: We use this wrapper-model greedy forward selection algorithm to select features for the classifiers on the second level of the hierarchy.

algorithm reaches its maximum accuracy after 10 iterations, several unnecessary features may be added which do not increase the accuracy, but may help find a better solution later on in the greedy process.

3.2 Combined Macro-class Selection

Figure 3 shows an expanded algorithm that is modified to select macro-classes as it iterates. In addition to the training data and label vector, it also requires the target accuracy threshold t, and the total number of classes n. The algorithm then outputs the selected feature set as well as L, a class map that assigns each class to one of p macro-classes, and C, the center of each macro-class in feature space.

The algorithm starts by selecting a moderate k for k-means clustering by taking the floor of $n/2$. It then tracks the selected feature set, the accuracy, and the corresponding outputs L and C. It then functions in the same iterative manner with the main difference being that the performance metric is based on the quality of the clustering at each iteration, not solely the classification accuracy as in the algorithm in Section 3.1.

The first step in the clustering process is to calculate the mean of each class in feature space (line 6 in Figure 3). The CMEAN function simply returns this set of points. These points are then clustered using a modified k-means algorithm, KMEANS2. This function clusters the input into a maximum of k clusters, where at each iteration, clusters that are empty or contain less than two points are automatically dropped. This liberal dropping scheme allows the algorithm to determine the number of clusters in a more unsupervised manner, rather than attempting to force k clus-

```
Input: $\mathbf{X} \in \mathbb{R}^{M \times N}, \mathbf{y} \in \mathbb{R}^M, t, n$
Output: $S, L, C \in \mathbb{R}^{p \times |S|}$
 1: $k = \lfloor n/2 \rfloor$
 2: $[A, S, L, C] \leftarrow [0, \emptyset, \emptyset, \emptyset]$
 3: while $|S| < N$ do
 4:     $[a, b, \psi, l, c] \leftarrow [0, 0, 0, \emptyset, \emptyset]$
 5:     for all $i \in \{1, \ldots, N\} \setminus S$ do
 6:         $\mu \leftarrow$ CMEAN($\mathbf{X}_{S \cup \{i\}}, \mathbf{y}$)
 7:         $\{c_i, p_i, q\} \leftarrow$ KMEANS2($\mu, k$)
 8:         $\{a_i, l_i\} \leftarrow$ KNN2($\mathbf{X}_{S \cup \{i\}}, c_i, q$)
 9:         if $A < t$ then
10:             $\psi_i \leftarrow (p_i > 1 \& \nexists \emptyset \in l_i \& a_i > A)$?$\Theta(a_i, p_i, l_i)$:0
11:         else
12:             $\psi_i \leftarrow (p_i > 1 \& \nexists \emptyset \in l_i \& a_i > t)$?$\Phi(a_i, p_i, l_i)$:0
13:         end if
14:         if $\psi_i > \psi$ then
15:             $[a, \psi, b, l, c] \leftarrow [a_i, \psi_i, i, l_i, c_i]$
16:         end if
17:     end for
18:     if $\psi = 0$ then
19:         break
20:     end if
21:     $[A, S, L, C] \leftarrow [a, S \cup \{b\}, l, c]$
22: end while
```

Figure 3: We use this modified algorithm to select features for the classifier on the top level of the hierarchy as well as determine the macro-classes that define the second level.

ters. KMEANS2 returns the cluster centers c_i, the number of clusters p_i, and a class map q.

The accuracy of the clustering is then determined by a modified k-NN algorithm, KNN2 (line 8). This function first runs a standard k-NN on $\mathbf{X}_{S \cup \{i\}}$, using c_i and q as a training data. KNN2 assigns each class to a macro-class cluster based on its popularity, forming a new class map l_i. This is done instead of maintaining q in an attempt to salvage good clusters that were not evident in the mean centers, but are in the actual training data. l_i and related accuracy a_i are returned.

Unlike the base algorithm in Section 3.1, our goal is not to simply maximize the accuracy. We aim to maximize the quality of the macro-classes while maintaining a "good enough" accuracy. We attempt this by building the algorithm to run in two phases. In the first phase (line 10), the feature with the highest score at each iteration is chosen as long as the accuracy is increased, emphasizing accuracy in the score equation Θ (Equation 1). If accuracy is not increased, the feature is disqualified by setting its score to zero. Once a certain accuracy target t is achieved, the algorithm continues to execute in the second phase (line 12). In this phase, the feature with the highest score is chosen as long as the accuracy is above the target threshold, de-emphasizing accuracy in a different score equation Φ (Equation 2). In both phases, clustering that results in less than two macro-classes or empty macro-classes is automatically disqualified.

$$\Theta = \frac{a_i^2 p_i^3}{\Gamma(l_i)^4} \tag{1}$$

$$\Phi = \frac{a_i p_i^3}{\Gamma(l_i)^4} \tag{2}$$

In the score equations, the Gamma function represents the frequency range of the class distribution of the given class map. For example, if l_i maps two classes to macro-class α, and six classes to macro-class β, then $\Gamma(l_i) = 6 - 2 = 4$. In general, the equations guide the algorithm to choose a feature set and macro-classes such that the accuracy and number of macro-classes is high, and the frequency range of the class distribution is low. In the ideal case, this should produce a fairly even class distribution with a high clustering accuracy of the training data.

The algorithm exits when either the current iteration disqualifies all candidate feature sets, or all features have been evaluated. We note that in actual implementation, we run this algorithm five times, using the result with the highest accuracy. This is to account for the randomness of k-means starting points. Traditionally, this is solved by using several "replicate" starting points in the k-means algorithm itself, choosing the clusters with the lowest within-cluster sums of point-to-centroid distances. We choose to rerun the entire algorithm because we are not necessarily interested in the best defined clusters, but rather how well they align with the training data.

3.3 Hierarchical Classification

The process for building the hierarchical classifier from the algorithms in Figures 2 and 3 is given in the following list.

1. Use the training data **X** and **y**, and an estimated target accuracy t with the algorithm in Section 3.2 to select features and macro-classes for the top level of the hierarchy.

2. Train a k-NN classifier using the selected features of **X** with the computed class map as the label vector, classifying the test sample into a macro-class.

3. Using the algorithm in Section 3.1, select features for each macro-class according to the class map.

4. Train a single k-NN for each macro-class, using only the training data corresponding to the macro-class's particular class set.

The process for classifying a test sample was given in Figure 1. Using the two-tier structure, the test sample is processed through two k-NN classification algorithms, the first to determine its macro-class, and the second to determine its final label.We note that although the added k-NN algorithm at the top level adds a second stage, the overall computational cost is reduced. This is because the k-NN algorithm at the top level is very inexpensive considering that the training set consists of only a single point per macro-class. At the second level, the dominating cost factor of the algorithm (calculating the test sample's distance from each of the training points) is divided with the training data between the mutually exclusive macro-classes.

4 EXPERIMENT SETUP

We test our method on human activity recognition using data collected from on-body IMU sensors. Computationally inexpensive features are computed from the data, and fed into the algorithms in Section 3 to form the hierarchical classifier. We then compare the results to those obtained using the non-hierarchical model built using only the base algorithm in Section 3.1. Our goal is to show that the macro-classes and features selected are good enough to improve the overall performance over the non-hierarchical model. In order for this to be achieved, there must be a high enough performance increase by using specialized features on each macro-class to justify the loss in accuracy by misclassifying data at the top level.

4.1 Datasets

We utilize two datasets: a subset of the Carnegie Mellon University Multimodal Activity (CMU-MMAC) Database (la Torre and Hodgins, 2009), and a dataset we collected from a smartphone. In both datasets, each IMU recorded instantaneous 3D acceleration (accelerometer), angular velocity (gyroscope), and orientation (magnetometer). An example of this data is given in Figure 4. Both datasets are about the same size, however the CMU-MMAC dataset contains more IMUs, resulting in a larger candidate feature pool. This is because features are computed across each dimension of each IMU. The CMU-MMAC dataset also contains significantly more classes, resulting in less average training data per class.

The full CMU-MMAC dataset consists of many subjects cooking a particular recipe in an unscripted manner while being observed by multiple sensors, including video cameras, IMUs, motion capture, and microphones. We use a subset of this data consisting of labeled data from 5 125Hz IMUs attached to the subjects arms, legs, and back. The subset contains 395 examples of 29 variable-length actions performed by 12 subjects cooking the brownie recipe. Labels were provided by the authors of (Taralova, 2009). The actions were chosen according to those used in (Spriggs et al., 2009) and (Fisher and Reddy, 2011). These actions were manually segmented out of the dataset, and all other activity was ignored. These actions are given in Table 1. The algorithms performance on unsegmented data is outside of the scope of this paper, and is the focus of our future work.

The smartphone dataset was collected for this paper. Each subject was given an Apple iPhone 4 loaded with the Sensor Data application and a piece of paper with the list of actions. The subject was then instructed to start the application, perform the action, stop the application, then write the index number next to the corresponding name on the labeling paper. Each action was recorded 5 times by 10 subjects using the single 60Hz IMU built into the phone. This resulted in 383 total action examples (not all subjects participated in each action). Once the data was recorded, we downloaded it according to the labels and manually trimmed each example to an 8.33 second clip for classification. We note that it is possible that the task scheduler on the phone may be accessing the sensor at a lower frequency, resulting in an inconsistent sample rate. The 9 actions are given in Table 2.

Table 1: A list of the actions in the CMU-MMAC dataset.

1. close-fridge	16. read-box
2. crack-egg	17. spray-pam
3. open-box	18. stir-bowl
4. open-cupboard1	19. stir-egg
5. open-cupboard2	20. switch_on
6. open-fridge	21. take-pan
7. pour-bowl-in-pan	22. take-egg
8. pour-bag-in-bowl	23. take-fork
9. pour-oil-in-bowl	24. take-oil
10. pour-oil-in-cup	25. take-pam
11. pour-water-in-bowl	26. twist_off-cap
12. pour-water-in-cup	27. twist_on-cap
13. put-pan-in-oven	28. walk–to-counter
14. put-oil-in-cupboard3	29. walk–to-fridge
15. put-pam-in-cupboard3	

Table 2: A list of the actions in the smartphone dataset.

1. Biking	6. Running
2. Climbing	7. Standing
3. Descending	8. Treadmill Walking
4. Exercise Biking	9. Walking
5. Jump Roping	

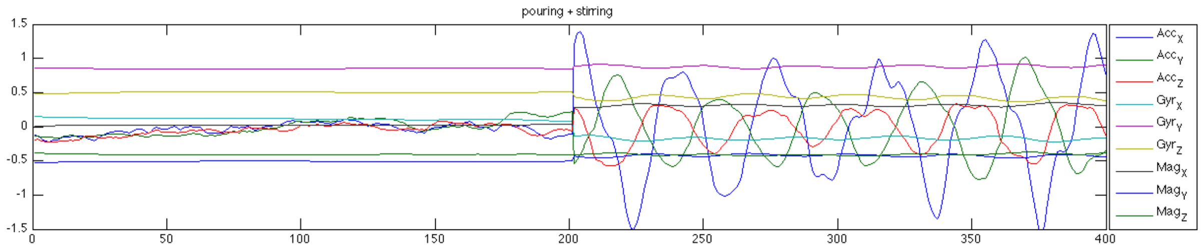

Figure 4: Each IMU in the dataset produces 9 data streams from a 3D accelerometer ($Acc_{X,Y,Z}$), Gyroscope ($Gyr_{X,Y,Z}$), and Magnetometer ($Mag_{X,Y,Z}$). This example shows the data stream from an IMU mounted on a subject's right arm while pouring and then stirring brownie mix. The data recorded during the transitions was removed from the dataset for our experiments.

These datasets are ideal for testing our method, mainly because the actions listed in Tables 1 and 2 are confusing because they are performed with similar movement. For example, in the smartphone dataset, the classifier must distinguish *walking* vs. *treadmill walking* as well as *climbing stairs* vs. *descending stairs* and *biking* vs. *exercise biking*. The CMU-MMAC dataset has more confusing action sets such as *walk-to-fridge* vs. *walk-to-counter* and *open-cupboard1* vs. *open-cupboard2*. The CMU-MMAC dataset also contains a larger number of classes.

4.2 Feature Calculation

We compute 13 variable-length statistical features across the 9 dimensions of each IMU sensor. These features, defined in Table 3, form 105 potential features for each IMU. For the CMU-MMAC dataset, this translates to 525 features for the 5 IMUs, and for the smartphone dataset, this translates to 105 features for the single IMU. In each of the formulas, X_i^j represents the ith data point of the jth dimension of the sensor X (accelerometer, gyroscope, magnetometer). We use these statistical features instead of the traditional frequency domain or PCA features because they are less computationally expensive to calculate, and have been proven to be effective activity recognition descriptors of IMU data in (Miluzzo et al., 2008), (Ermes et al., 2008), and (Karantonis et al., 2006).

4.3 Testing Procedure

We test each dataset according to the hierarchical classification procedure listed in Section 3.3. Leave-one-subject-out cross validation is used in order to test the method on each subject independently, excluding that subject's data from the training data used to build the model. The results of each subject are concatenated to calculate the final accuracy across the entire dataset. The target clustering accuracy value t is selected to be 90% and 95% for the CMU-MMAC and smartphone datasets respectively. A lower target accuracy is used for the CMU-MMAC dataset because of the larger number of classes.

Table 3: Statistical features calculated from the IMU data.

Feature	**Size**	**Formula**
Mean	9	$\mu_{X^j} = \frac{1}{\ell}\sum_{i=1}^{\ell} X_i^j$
Variance	9	$\sigma_{X^j}^2 = \frac{1}{\ell}\sum_{i=1}^{\ell}(X_i^j - \mu)^2$
Minimum	9	$min_{X^j} = \text{minimum}(X_i^j)$
Maximum	9	$max_{X^j} = \text{maximum}(X_i^j)$
Range	9	$range_{X^j} = max_{X_i^j} - min_{X_i^j}$
Mean Crossing Rate	9	$mcr_{X^j} = \frac{1}{\ell-1}\sum_{i=\ell}^{N-1}\Upsilon\{(X_i^j - \mu_{X^j})(X_{i+1}^j - \mu_{X^j}) < 0\}$, where Υ is the indicator function
Root Mean Square	9	$rms_{X^j} = \sqrt{\frac{1}{\ell}\sum_{i=1}^{\ell} {X_i^j}^2}$
Skew	9	$skew_{X^j} = \frac{1}{\ell\sigma_{X^j}^3}\sum_{i=1}^{\ell}(X_i^j - \mu_{X^j})^3$
Average Entropy	9	$H_{X^j} = -\frac{1}{\ell}\sum_{i=1}^{\ell} p(X_i^j)\log(p(X_i^j))$
Kurtosis	9	$kurt_{X^j} = \frac{1}{\ell\sigma_{X^j}^4}\sum_{i=1}^{\ell}(X_i^j - \mu_{X^j})^4$
Correlation	9	$corr_{X^{ab}} = \frac{1}{\ell\sigma_{X^a}\sigma_{X^b}}\sum_{i=1}^{\ell}(X_i^a - \mu_{X^a})(X_i^b - \mu_{X^b})$, for [a,b]={[1,2],[1,3],[2,3]}
Average Magnitude Area	3	$SMA_X = \frac{1}{\ell}\sum_{i=1}^{\ell}(\lvert X_i^1\rvert + \lvert X_i^2\rvert + \lvert X_i^3\rvert)$
Average Energy Expenditure	3	$EE_X = \frac{1}{\ell}\sum_{i=1}^{\ell}\sqrt{{X_i^1}^2 + {X_i^2}^2 + {X_i^3}^2}$

5 RESULTS

For each dataset, we present the total classification accuracy of the hierarchical model compared to the non-hierarchical baseline. The top-level clustering accuracy is also given in order to indicate how well the macro-classes selected from the training data were able to be generalized to the testing data, recalling that top-level classification accuracy significantly impacts performance in a hierarchical classifier since it is essentially the maximum achievable total accuracy.

In addition to classification results, we also present a novel 2D histogram matrix to show the gist of the macro-classes selected for each dataset. The matrices shown in Figures 6 and 7 visualize how often each class is grouped into the same macro-class as another after the algorithm has run on all of the subjects. Each row corresponds to a class, and the intensity of the marking at each corresponding column represents the frequency in which the row class was grouped into the same macro-class as the column class. For example, the first row in Figure 6 indicates that action 1 is always grouped into the same macro-class as actions 4, 6, 12, 17, 22, 23, 28, and 29 since the marks at these columns of row 1 are completely black. Action 5 is grouped into the same macro-class as action 1 about 60% of the time, indicated by the gray mark in column 5.

In addition to visualizing the gist of the selected macro-classes, the histogram matrix can also visually depict the quality and nature of the selected-macro classes. The quality of the macro-class selection is indicated by the ability of the algorithm to group the same classes into a macro-class regardless of which subjects the algorithm is trained on. This can be seen in the previous example in which the column classes with completely black marks in row 1 were always grouped together regardless of the training set used during cross validation. In general, we can say that a matrix consisting of mostly black or white marks is of good quality because the macro-classes are well defined across different training sets with close to 100% or 0% matching. Additionally, if the classes are listed in such a way that the naturally similar classes are adjacent, dark clusters will form if the naturally similar classes are generally grouped into the same macro-class. This is further explained in Section 5.1.

In order to aid future feature selection research, we also review which features from Table 3 were selected when using each model.

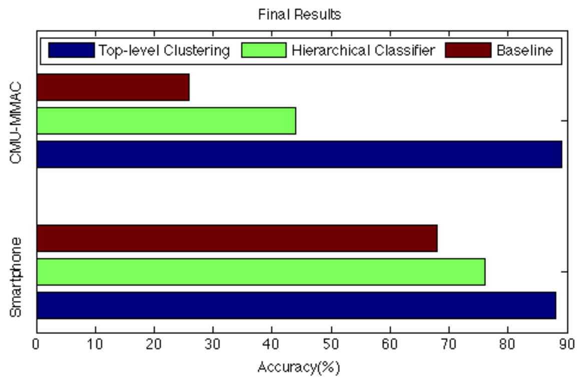

Figure 5: The final results show that the hierarchical classifier built using the algorithms presented in this paper outperforms the non-hierarchical baseline in both datasets. The high top-level clustering accuracy in both datasets indicates the high quality of the selected macro-classes.

5.1 CMU-MMAC Results

The classification results for the CMU-MMAC dataset are given in Figure 5. The classification accuracy using the hierarchical model was 44%, a 69% improvement over the non-hierarchical baseline of 26%. The top-level clustering accuracy was 89%.

The histogram matrix in Figure 6 shows the gist of the macro-class selection. Most of the graph is completely white or very dark, indicating good quality macro-classes. In this matrix, the classes are listed in lexicographical order, making naturally similar actions beginning with the same verb (e.g. *pour*-oil-in-bowl, *pour*-water-in-bowl) adjacent to one another. In this way, it can be seen that some of the macro-classes correspond to naturally similar classes. Specifically, the pouring actions in rows 7-11 are always grouped together with the other pouring actions in columns 7-11, the stirring actions in columns 18-19, and the twisting actions in columns 26-27. Theses groups are emphasized by colored ellipses in Figure 6. The average number of macro-classes created from the 29 base classes was 4.

5.2 Smartphone Results

The classification results for the smartphone dataset are also given in Figure 5. The total classification accuracy using the hierarchical model was 76%, a 12% improvement over the non-hierarchical baseline of 68%. The top-level clustering accuracy was 88%.

The histogram matrix for the smartphone dataset is given in Figure 7. Like the CMU-MMAC dataset, most of the marks on the graph are completely white or very dark, indicating good, subject-independent macro-classes. We can also see that the intersec-

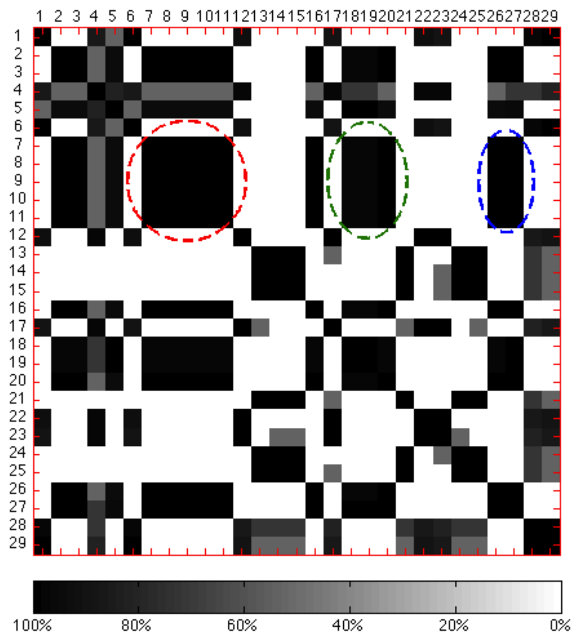
Figure 6: The histogram matrix for the CMU-MMAC dataset. The red, green, and blue ellipses highlight how the pouring actions are generally grouped with the other pouring actions, the stirring actions, and the twisting actions respectively. The index numbers correspond to the actions listed in Table 1.

tions of naturally similar classes such as *climbing/descending* and *running/walking* are filled with completely black marks, indicating that these classes were grouped together every time. The average number of macro-classes created from the 9 base classes was 2.6.

5.3 Feature Selection Results

The distribution of the features selected is given in Figure 8. In general, the overall most useful features were mean, variance, and entropy, having the highest distribution in both models. The least useful features were the correlation and signal magnitude area, having the lowest distribution in both models. We also note that the standard deviation of the distribution of the hierarchical model is 5.6, which is less than the 6.8 of the non-hierarchical model. This implies that the features were more evenly distributed in the hierarchical model. This is expected considering that features that are less descriptive overall are eliminated in feature selection for the non-hierarchical model, but can be used as specialized features in one of the hierarchical model's subclassifiers. This is specifically evident in the mean crossing rate and kurtosis features, both of which more than doubled their representation in the hierarchical model.

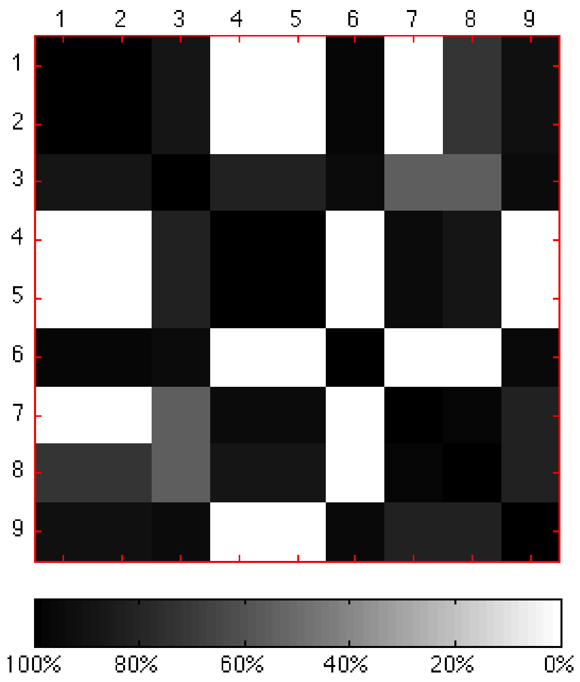
Figure 7: The histogram matrix for the smartphone dataset. The index numbers correspond to the actions listed in Table 2.

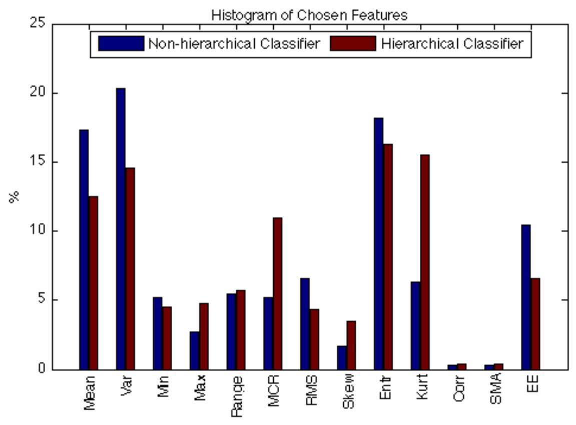

Figure 8: Histogram showing the distribution of the selected features across both datasets for the non-hierarchical and hierarchical models. The bottom labels correspond to the features listed in Table 3.

6 CONCLUSION

The results show that our algorithm performs well in selecting macro-classes and features for hierarchical classification, as accuracy was improved in both datasets over the non-hierarchical baseline. We note that with the exception of one subject, the hierarchical classifier either matches or outperforms the non-hierarchical baseline for every individual subject. This empirically shows that the macro-classes and features selected by our algorithm are useful in creating the hierarchical k-NN classifier. We emphasize that the improvement was much greater in the more complex CMU-MMAC dataset (69% vs. 12%). This is because the hierarchical classifier was built to group and handle similar classes separately with specialized features. Therefore, the more confusing dataset yields a higher improvement.

However, we do recognize that although we outperform the non-hierarchical baselines, the resulting accuracies are still low compared to previous work in (Fisher and Reddy, 2011). This is because, instead of focusing on the maximization of total accuracy as in previous work, we focus on generating quality macro-classes and testing the performance impact of using the respective specialized feature sets. In an effort to minimize the computational cost of our resulting algorithm, we use computationally inexpensive statistical features and k-NN classification on the second level of the hierarchy. Our accuracy would most likely be substantially improved at the cost of computational resources by using the more complex features and classification methods of previous work on the second level of the hierarchy. Once the test sample has been correctly classified into a macro-class at the top level (which we achieve very high performance), we note that any type of feature set or classifier can be used by the subsequent classification nodes.

Overall, we contribute a new algorithm to improve the performance of the k-NN classifier by building a hierarchical classification model with specialized feature selection. Our results show significant improvement over the baseline, with the possibility to improve further by using more complex features or classifiers on the bottom level of the hierarchy.

ACKNOWLEDGMENTS

Data used in this paper was obtained from kitchen.cs.cmu.edu and the data collection was funded in part by the National Science Foundation under Grant No. EEEC-0540865.

REFERENCES

Ermes, M., Parkk, J., Mantyjarvi, J., and Korhonen, I. (2008). Detection of daily activities and sports with wearable sensors in controlled and uncontrolled conditions. *IEEE Transactions on Information Technology in Biomedicine*, 12(1).

Fisher, R. and Reddy, P. (2011). Supervised multi-modal action classification. Technical report, Carnegie Mellon University.

Ganti, R., Srinivasan, S., and Gacic, A. (2010). Multisensor fusion in smartphones for lifestyle monitoring. In *Proceedings of 2010 International Conference on Body Sensor Networks*.

Iso, T. and Yamazaki, K. (2008). Gait analyzer based on a cell phone with a single three-axis accelerometer. In *Proceedings of 6th ACM Conference on Embedded Networked Sensor Systems*.

Karantonis, D., Narayanan, M., Mathie, M., Lovell, N., and Celler, B. (2006). Implementation of a real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring. *IEEE Transactions on Information Technology in Biomedicine*, 10(1).

Kim, K.-J., Hassan, M. M., Na, S., and Huh, E.-N. (2009). Dementia wandering detection and activity recognition algorithm using tri-axial accelerometer sensors. In *Proceedings of the 4th International Conference on Ubiquitous Information Technologies & Applications*.

Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artificial Intelligence*, 97(1-2).

la Torre, F. D. and Hodgins, J. (2009). Guide to the carnegie mellon university multimodal activity (cmu-mmac) database. Technical Report CMU-RI-TR-08-2, Carnegie Mellon University.

Law, M., Figueiredo, M., and Jain, A. (2004). Simultaneous feature selection and clustering using mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9).

Liu, H. and Yu, L. (2005). Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on Knowledge and Data Engineering*, 17(4).

Miluzzo, E., Lane, N., Fodor, K., Peterson, R., Lu, H., Musolesi, M., Eisenman, S., Zheng, X., and Campbell, A. (2008). Sensing meets mobile social networks: The design, implementation and evaluation of the cenceme application. In *Proceedings of 6th ACM Conference on Embedded Networked Sensor Systems*.

Saponas, T. S., Lester, J., Froehlich, J., Fogarty, J., and Landay, J. (2008). ilearn on the iphone: Real-time human activity classification on commodity mobile phones. Cse technical report, University of Washington.

Spriggs, E., la Torre Frade, F. D., and Hebert, M. (2009). Temporal segmentation and activity classification from first-person sensing. In *Proceedings of IEEE Workshop on Egocentric Vision at Conference on Computer Vision and Pattern Recognition*.

Talavera, L. (2005). An evaluation of filter and wrapper methods for feature selection in categorical clustering. In *Proceedings of 6th International Symposium on Intelligent Data Analysis*.

Taralova, E. (2009). Cmu multi-modal activity dataset annotations. In *http://www.cs.cmu.edu/ espriggs/cmu-mmac/annotations/*.

Wang, Y.-C. F. and Casasent, D. (2008). New support vector-based design method for binary hierarchical classifiers for multi-class classification problems. *Neural Networks*, 21(2-3).

Zeng, H. and Cheung, Y.-M. (2009). A new feature selection method for gaussian mixture clustering. *Pattern Recognition*, 42(2).