Energy-Efficient Activity Recognition using Prediction

Dawud Gordon, Jürgen Czerny, Takashi Miyaki, Michael Beigl
Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: [Firstname.Lastname]@kit.edu

Abstract

Energy storage is quickly becoming the limiting factor in mobile pervasive technology. For intelligent wearable applications to be practical, methods for low power activity recognition must be embedded in mobile devices. We present a novel method for activity recognition which leverages the predictability of human behavior to conserve energy. The novel algorithm accomplishes this by quantifying activity-sensor dependencies, and using prediction methods to identify likely future activities. Sensors are then identified which can be temporarily turned off at little or no recognition cost. The approach is implemented and simulated using an activity recognition data set, revealing that large savings in energy are possible at very low cost (e.g. 84% energy savings for a loss of 1.2 pp in recognition).

1 Introduction and Related Work

As concepts from pervasive and mobile computing become more mainstream, the community seeks practical approaches for realizing pervasive technology. Situational, context or activity recognition techniques provide a method for machines to recognize human and social situations, allowing them to act proactively without contradicting or offending their owners. Modern technological devices such as smart phones or wireless sensor networks are now able to handle these algorithms [15] as processing power and memory improve over time according to Moore's Law.

Energy storage in such devices is not subject to the same doubling effects and is quickly becoming the limiting factor in pervasive technology. This can be seen clearly when reviewing the battery lifetimes for mobile phones over the past 10 years. The cost of communication in terms of energy consumption is another factor which does not scale according to Moore's Law, indicating that for intelligent wearable applications to be practical, methods for low power situational recognition must be embedded in mobile devices.

Many things we do have a certain repetitiveness or periodicity about them [17], and are therefore predictable to a certain extent. This information can be used to improve recognition abilities [11]. It is the proposal of this work that it can also be used to reduce the power consumption of the recognizing device as well. The idea is simple, given a scenario where activities are performed in a manner which is predictable, probable future activities can be forecast. Sensors which are not needed to decipher probable activities from each other can be turned off, conserving energy without greatly impacting recognition rates.

Embedded classification for mobile devices is not a new concept and goes as far back as 1997 [4], where Bouten el al. used simple signal processing to measure activity levels of users wearing a mobile device. Several methods for low power embedded context classification have been introduced in the activity and context recognition community. Cacmakci et al. [5] and Stäger et al. [14] introduce straightforward approaches to low power recognition of contexts and activities in embedded systems using inertial or audio sensors respectively. Krause et al. [9] propose to dynamically reduce sensor sampling to conserve energy, thereby greatly increasing the lifetime of a single battery. A similar approach was presented by Sun et al. [16] where coarse-grained activity levels were locally recognized to adjust the sensor sample rate. Benbasat et al. [1] introduce a method for conserving energy in a system with redundant sensors which are switched on and off dynamically based on the level of activity currently measured. Roy et al. [12] use sensor configurations which are selected for specific activities based on the minimum requirements of an application. The result of the research discussed here is that there is always a trade off between how well activities can be recognized, and how much energy it costs to do it [15, 2].

Here we present a different method for using context (or activity) prediction to conserve energy. Context prediction is the process of using context history to predict contexts, situations or activities which will occur in the future [10]. This can be done at several different abstraction levels [13], ranging from extrapolating raw sensor data into the future, to predicting abstract concepts such as activities.

This work proposes that by leveraging the predictability of human actions, it is possible to tip the balance of the energy/recognition trade off to conserve energy resources. This approach was first proposed in a poster [7], and is independent of the classification and prediction algorithms used. The performance is simulated using a preexisting activity recognition data set [6], where artificial data sets are generated in order to evaluate different scenario parameters. The results indicate that the novel approach allows for application and scenario specific selection of an optimal recognition/energy trade off, producing large energy savings, even for small recognition losses (e.g. 84% energy savings for 1.2 pp loss in recognition).

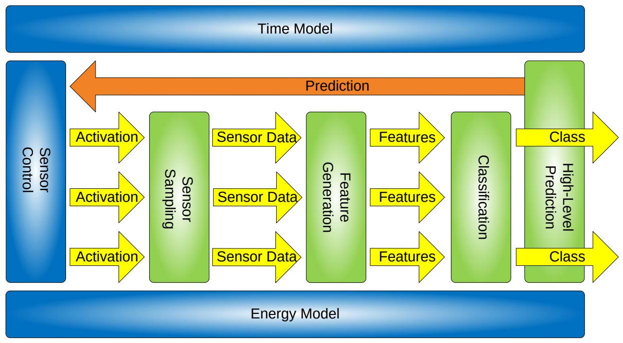

Figure 1. The Novel Algorithm (Green) and Simulation Environment (Blue)

2 Algorithmic Approach

The standard process for activity recognition using machine learning algorithms is straightforward. Sensors are sampled in parallel at an arbitrary but constant rate and period of time. The data is then saved as a discrete multi-dimensional vector, referred to as a sample window. This window is processed using different algorithms to generate signal features, e.g. standard deviation, average, FFT or cepstral coefficients. Which features are used depends on the application, i.e. which activities we want to recognize, and the type of sensors being used, and are referred to all together as a feature vector. A machine learning algorithm is given the task of recognizing which activity was occurring during the sample window, based on its feature vector.

We propose integrating prediction into the process to improve energy consumption as demonstrated in Fig. 1. First, *activated* sensors are sampled to generate a sample window. The sample window is then processed into a feature vector, and *classified* as to which activity is being performed. Based on the classification history, future activities which are likely to occur are *predicted*. An appropriate sensor configuration is then activated to distinguish only the *likely activities*, and the process repeats itself.

During the course of this research, three parameters have been identified which affect the trade off between energy and recognition. The first is the **predictability $\boldsymbol{\kappa}$** of the sequence of activities, or the inherent predictability of the scenario itself. A low value for κ indicates that prediction results are no better than random, where a $\kappa = 1$ indicates a 100% prediction accuracy. In real world scenarios, κ simply equates to the prediction rate for a given predictor and scenario. This parameter cannot be influenced by the designer, and can only be quantified by analyzing the scenario and predictor beforehand. The second parameter affecting performance is **$\boldsymbol{\rho}$, the number of classes which are predicted** at each time step. The more classes which are predicted, the better the chance that the next class is actually among the predicted classes (correct prediction), but the lower the savings will be as the system accounts for more possible activities. Therefore ρ specifies the level of risk, which allows the designer to tip the odds towards recognition or energy as will be seen later. The third parameter is application specific, and is referred to as the **loss parameter $\boldsymbol{\lambda}$**, which specifies the amount of recognition which can be sacrificed in order to conserve energy without breaking the application's requirements. A λ value of 0 indicates that optimizations causing any loss at all, however minimal, are not acceptable, and $\lambda = 1$ means energy savings are of the utmost priority, and recognition rates are of no importance.

2.1 Weighting Sensors to Activities

Now the method for selecting which sensors to activate based on predicted activities will be explained. When observing the chain of events in the context classification process, each feature in the set of features used $f \in F$ is implicitly mapped onto a single sensor in the set of sensors $s \in S$. That sensor generates the data for this feature, producing the surjective mapping a of features onto sensors $a\colon F \to S$. Reversely, each sensor s_i is then "responsible" for a subset of features $\widetilde{F}_{s_i} \in F$, meaning the features in $\widetilde{F}_{s_i}$ are generated over data from sensor s_i.

Mapping activity classes onto the sensors over the features is not as simple. This mapping cannot be carried out independent of the classifying algorithm, as each algorithm has a different method of measuring the distance between two vectors [3]. An overview of selecting features which best suite an embedded application is presented by Könönen et al. [8], providing a *sensor to application* mapping. A method for generating a *sensor to class (activity)* mapping by relevance or importance was proposed by Roy et al. [12], which they referred to as quality-of-inference (QoINF). As will be discussed later, this method is not effective for the approach and data set presented here.

Turning sensors on and off will result in a dynamic fea-

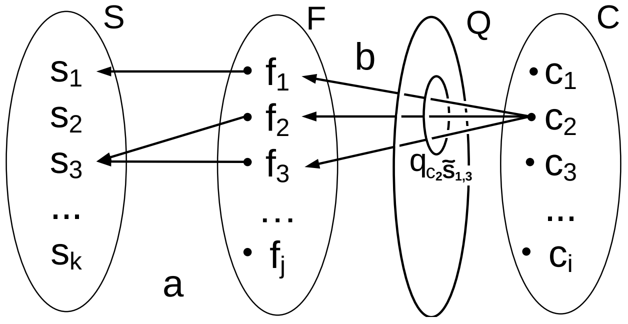

Figure 2. Features (F) from Sensors (S) s_1 and s_3 have Value (Q) $q_{c_2\widetilde{S}_{1,3}}$ for Class (C) c_2

ture vector length, and for this reason we will consider classifiers which can natively support this. Specifically, nearest-neighbor classifiers are well suited to this task as omitting a feature represents a dimensional reduction of the labeled training vector space, and the missing features are simply excluded from the distance calculation. Probabilistic models are also well suited as the observational distributions for missing variables can be ignored when calculating the probabilities of the hidden states. Both examples lose only the information that would have been gained from the missing features, but are not further negatively affected [3].

In order to generate the weighted mapping (the weight is the dependency of activities on sensors), training data is gathered for each class. Weight calculation was done by testing the trained classifier against all training vectors for each class and simulating different feature combinations. Selected features were turned off and the dependency of each class on those features was evaluated. The degree of dependency is the drop in accuracy compared to the full feature vector: a large drop in recognition indicates a high dependency, a small drop, low dependency.

Initially the intent was to only evaluate the weight for each feature individually. The cost/dependency weights for a sensor could then be calculated by summation of the weights of its features, assuming $q_{cf_i} + q_{cf_j} \approx q_{cf_{ij}}$, or that the cost of turning off two features, is the cost of the one plus the cost of the other as indicated by Roy et al. [12]. This however proved to be too inaccurate to be useful due to the conditional dependence of features and sensors, making $q_{cs_i} + q_{cs_j} \leq q_{cs_{ij}}$ [8]. Therefor, Q values were calculated for each class against all possible sensor subsets directly, instead of by summing single feature or sensor values.

In order to correctly estimate the optimal sensor subset $\widetilde{S}$ for a sensing and classification step, the matrix Q must be calculated only once at training time. The resulting mappings can be seen in Fig. 2, showing one mapping of class $c_2 \in C$ onto sensors $\widetilde{S}_{1,3} \in S$ over features generated from $\widetilde{S}_{1,3}$. Each mapping in $b\colon C \xrightarrow{Q} F$ represents one element in the Q matrix, in this case $q_{c_2\widetilde{S}_{1,3}} \in Q$. The Q matrix is indexed by the power set of S without the empty set, or $\widetilde{S} \in \wp(S)\backslash\{\}$, and the classes $c \in C$, resulting in a $|C| \times (2^{|S|} - 1)$ matrix. The value at each point i, j indexed by c_i and $\widetilde{S}_j$ is the recognition loss when classifying c_i using sensor subset $\widetilde{S}_j$ compared to using all sensors S over a set of evaluation data samples. Now, for each class $c_i \in \widetilde{C}_{t+1}$ where $\widetilde{C}_{t+1}$ is the set of activities predicted to occur at the next time step, a set of sensors $\widetilde{S}_{t+1}$ can be identified which is optimal with respect to λ. This is accomplished by selecting the sensor subset $\widetilde{S}$ for the next period $t+1$ such that it fulfills Eq. (1).

$$\widetilde{S}_{t+1} = \underset{En(\widetilde{S})}{\arg\min}, \forall_{c \in \widetilde{C}_{t+1}} q_{\widetilde{S},c} \leq \lambda \tag{1}$$

Where $En(\widetilde{S})$ is the combined energy consumption of all sensors in $\widetilde{S}$. Simply put, in order to recognize the classes predicted to occur next $\widetilde{C}_{t+1}$, the sensor configuration $\widetilde{S}$ is selected which saves the most energy $En(\widetilde{S})$ without violating the acceptable loss parameter λ for *any* of the predicted activities $c \in \widetilde{C}_{t+1}$. Effectively, this selects the sensor configuration with the lowest energy consumption that is still capable of recognizing the predicted classes, while maintaining acceptable recognition rates. The next section will analyze the use of context prediction to generate a set of classes which are likely to appear in the next sample window ($\widetilde{C}_{t+1}$).

2.2 Context Prediction

Context prediction is used to estimate a subset of all contexts or activities $\widetilde{C}_{t+1} \in C$ which are most likely to occur at the next time step $t+1$. The cardinality of $|\widetilde{C}_{t+1}| = \rho$ is a parameter which can be adjusted, and allows the designer to select the recognition accuracy risk against the energy reward as will be shown in Sec. 4. This approach is independent of the algorithm or abstraction level used for prediction. Important is only the quantification of the predictability parameter κ which is simply an indicator of how well the predictor is able to forecast the given scenario (predictor accuracy). The results presented here should therefore still apply for all scenarios and prediction algorithms.

As indicated by Fig. 1, high-level context information at the activity or context abstraction level is used for prediction. Using low-level, sensory or feature data is also an option, but high-level prediction reduces complexity in terms of training and classification [13]. The algorithm used for prediction is a first-order Markov chain consisting of states $c \in C$. At each time step, the probability $P(c_{i,t+1}|c_t)$ for each $c_i \in C$ is calculated, and the ρ states with the highest probabilities are output as predictions.

3 Implementation and Simulation

This section presents the algorithmic implementation and the simulation environment. Both were programmed using the Python programming language.

3.1 Simulation Environment

The main concept is to leverage the predictability of human actions in order to conserve a large amount of energy while only sacrificing a small amount of recognition capabilities. The simulation environment was designed to evaluate the method for various degrees of predictability κ.

The data set used for evaluation [6] contains 142 minutes of data from 4 sensors (see Tab. 1), sampled from 5 subjects performing 8 activities (taking a bus, riding a bike, walking, jogging, taking the elevator, typing at a desk, going up/down stairs, and standing). The system simulates real time through a replay mechanism using the recorded data. The data set is cut up into one second windows without overlap, over which features are generated. The resulting feature vectors are then fed to the novel algorithms as if it were being generated in real time. The sensor configurations are simulated, where for a specific sensor configuration $\widetilde{S}$, the features $\widetilde{F}_{\widetilde{S}}$ are present in the feature vector and all others are omitted.

Once a sensor configuration $\widetilde{S}$ has been selected, the features $\widetilde{F}_{\widetilde{S}}$ are calculated. Per sensor, the following features are calculated: average, standard deviation, area under the curve, min-max difference, Shannon entropy, and FFT peak as these are common and effective features [6].

The energy consumed by the device $En(\widetilde{S})$ is recorded for the time step. The total consumption consists of the consumption of each sensor, as well as the energy consumption of the microprocessor during the course of the sample window, or one second. The energy model is simplistic, ramp-up and ramp-down times/consumptions of the sensors are not modeled, and the processor consumption is modeled as being constant regardless of load. This approximation does not account for the added load of prediction, but the method used here has a computational load of only $\mathcal{O}(|\mathcal{C}|)$ [7]. The energy consumption rates for each device simulated can be found in Tab. 1. At each time step, a new $\widetilde{S}_t$ is provided by the algorithm, which results in a different feature vector consisting of features $\widetilde{F}_{\widetilde{S}}$, and a different energy cost. The amount of energy consumed can then be compared with the amount consumed for the reference case when $\widetilde{S} = S$, i.e. all sensors remain on, for comparison.

As with energy consumption, the simulation environment also records classification results, both for the novel algorithm and for the reference case. For each time step, the algorithm classifies $\widetilde{F}_{\widetilde{S}}$ and the result of the classification is recorded, along with the energy consumption. At the same time, the complete feature vector F_S, consisting of the entire feature set F is also classified and the result is stored for comparison with the reference system. The result is that the simulator records the energy consumption and classification results for both the novel prediction-based activity recognition algorithm, as well as the reference case when all sensors remain on.

Table 1. Component Energy Consumption [6]

Element		Energy Cost (mW)	
Function	Name	Online	Offline
Light	APDS-9003	8.25	0.0
Acceleration	ADXL335	1.4	0.0
Temperature	TC1047	0.1155	0.0
Microproc.	PIC18LF14K	0.0512	0.0
Vibration	MVS 0608.02	0.0015	0.0

Artificial Data Set In order to evaluate the behavior of the system for different degrees of predictability (κ), artificial data sets are generated using the original data set and a generative probabilistic model shown in Fig. 3. The goal is to generate a data set which is predictable to a specified degree by the predicting algorithm, meaning that it results in a certain prediction accuracy. A Markov chain assumes that the process being modeled holds with the Markov property. It follows that by changing how pronounced the Markov property is in the data, the accuracy of the predictor can be set. The predictability is defined as $\kappa \in [\frac{1}{|C|}, 1]$ where a value of 1 indicates that

$$\forall i \exists j | P(c_{t+1} = c_j | c_t = c_i) = 1$$

and a value of $\frac{1}{|C|}$ indicates that

$$\forall i, j | P(c_{t+1} = c_j | c_t = c_i) = \frac{1}{|C|}$$

or that all transitions are equally likely. Setting $\kappa = \frac{1}{|C|}$ is the lower bound for predictability, as there are $|C|$ transitions leaving each state, and the probabilities of all exit transitions sum to one. Assigning κ a lower value than this means at least one exit transition must have a probability higher than $\frac{1}{|C|}$, increasing predictability.

Using κ, we can generate a HMM (not to be confused with the HMM used for recognition) by ordering states such that each state has 1 and only transition to a different state with probability κ, and only 1 transition from a different state to itself with probability κ. All other transitions have probability $\bar{\kappa} = \frac{1-\kappa}{|C|-1}$. Simply put, as κ approaches 1, the state following the current state becomes more and more certain, and therefore easier to predict. As κ approaches the lower bound of $\kappa = \frac{1}{|C|}$, the next state becomes more random, and harder to predict. Once this model is created, traversing it generates emissions which are sample windows from the original data set for the given activity. This is demonstrated in Fig. 3 for an example 3-class dataset.

3.2 Experimental Process

The algorithm presented here is not application specific. It is meant to reduce the cost of embedded activity and context recognition in scenarios with repetitive temporal patterns. Each application is different in terms of the optimal trade off between energy consumption and accuracy [15]. The algorithm and the following evaluation is conducted without a specific cost model, but allows the reader to evaluate the effectiveness for the application scenario at hand.

The classifiers used are the Hidden Markov Model [11] (HMM), and the k-Nearest-Neighbors (kNN) [3] algorithms, as they are both easily adapted to a variable feature vector length, and have both been evaluated on the data set previously [6]. The algorithm requires two separate sets of training data, one to train the classifier and predictor, and a separate one to populate the Q matrix using the trained classifier. A third data set is required for evaluation.

Training Phase Each artificial data set is separated into 3 portions, the data used to train the classifier and predictor $\widetilde{D}_{Train}$ makes up 60% of the original data set D. Another 20% $\widetilde{D}_Q$ is used to calculate the Q matrix, as using $\widetilde{D}_{Train}$ for this purpose results in overfitting, and therefore distorted loss values in Q. Finally, the last 20% $\widetilde{D}_{Eval}$ is used to evaluate the performance of the whole system, and in this experiment contains 3595 sample windows in total.

In the first step, $\widetilde{D}_{Train}$ is used to train the classifier, either HMM or kNN, as well as the Markov chain used for prediction. In this phase prediction is not used for classification, and both the classifiers and the predictor are trained on all features of F. In the second training step, $\widetilde{D}_Q$ is used to populate the Q matrix by evaluating the recognition rate of every class c_i with every permutation of $\widetilde{S}$. Therefore, every combination in $C \times \wp(S) \backslash \{\}$ is evaluated in a separate classification phase, using all vectors for activities of the current subset.

Testing Phase In the testing, or evaluation phase, the classifier algorithms are run on $\widetilde{D}_{Eval}$ in parallel. At each classification time step, the $\widetilde{S}_t$ resulting from the previous time step is used to generate a new feature vector $\widetilde{F}_t$. This vector is then classified, either by the HMM or kNN classification algorithm. Based on this classification, the prediction algorithm predicts ρ probable classes $\widetilde{C}_{t+1}$ for the next time step. Next, the sensor subset $\widetilde{S}_{t+1}$ is selected such that it

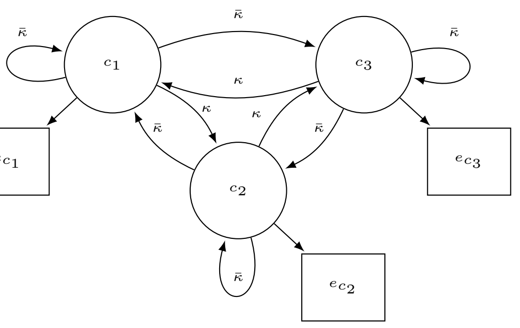

Figure 3. Generative Model for Constructing an Artificial Data Set with 3 Classes (C), Emissions (E) and Predictability κ

fulfills Eq. (1). At the end of each step the simulation environment records the classification result using F, $\widetilde{F}_t$, the ground truth for that sample window, the sensor subset $\widetilde{S}$, the energy consumed $En(\widetilde{S}_t)$ and the predictions $\widetilde{S}_{t+1}$ for the next time step.

4 Evaluation

For each different degree of scenario **predictability** κ (12.5:87.5 step 12.5), a different artificial data set was generated. The **number of predicted states** ρ (1:8 step 1), the **acceptable loss parameter** λ (0:1 step 0.1) and the classifier (HMM and kNN) were permuted to evaluate the output parameters over each data set. The results are 6-dimensional, consisting of dimensions ρ, λ and κ, the classifier, recognition rates and energy consumption. It is impossible to impart this information in its entirety in the space allotted, therefor major insights will be detailed and demonstrated with graphical excerpts.

4.1 Recognition Loss

For a given loss parameter λ, loss of recognition decreases monotonically (meaning recognition increases) for an increasing ρ (number of states predicted). This is demonstrated by Fig. 4 for a κ of 0.125, and again in Fig. 5 for a κ of 0.875 for both the HMM and kNN classifiers. This is again evident in Fig. 7, where for a given λ, increasing ρ either reduces or leaves recognition loss unchanged. In other words, increasing the number of classes predicted increases recognition rates and reduces energy savings.

The same also applies to the acceptable loss λ, where for a given classifier and ρ, loss in recognition and energy savings increase monotonically with λ. The implication is that the acceptable loss parameter λ does indeed function as an indicator for how much loss can be sacrificed as proposed.

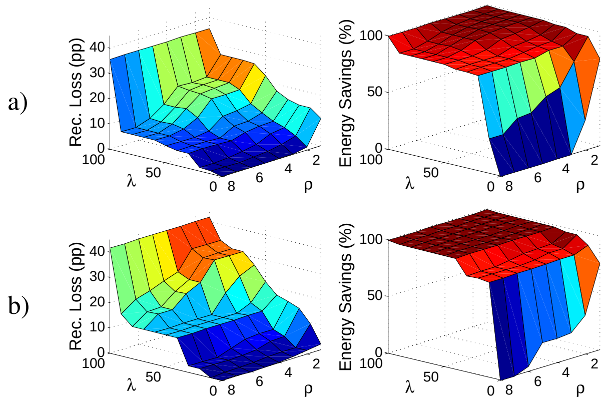

Figure 4. Recognition Loss and Energy Savings for the HMM (a) and kNN (b), κ=0.125

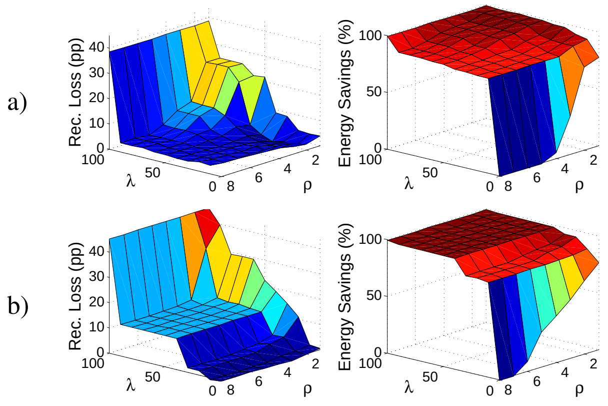

Figure 5. Recognition Loss and Energy Savings for the HMM (a) and kNN (b), κ=0.875

The monotonic behavior of ρ and λ implies that for a given predictability κ, the lowest recognition loss (best recognition) is obtained by $\rho = |C|$ and $\lambda = 0$, and the highest loss (worst recognition) when $\rho = 1$ and $\lambda = 1$.

Observing accuracy loss over κ for fixed values of λ and ρ is not as clear cut. In Fig. 6, varying κ affects recognition for $\rho = 4$ using the HMM, where recognition loss is not monotonically decreasing as κ increases (compare $\kappa = 0.125$ with $\kappa = 0.25$ for $\lambda = 0.8$), but the trend is decreasing. For the kNN classifier, the effects of κ are minimal when compared to the HMM as seen in Figs. 4 - 7.

4.2 Energy Consumption Rates

The energy savings is defined as the relative decrease in energy consumed over the evaluation of $\widetilde{D}_{Eval}$ between the reference classifier with all sensors on and the novel algorithm. When observing Fig. 4 and Fig. 5, the acceptable loss parameter λ has a far greater influence on energy savings than either ρ or κ. Fig. 6 and Fig. 7 demonstrate this by showing very little differentiation in energy savings for either κ or ρ respectively. In all cases, a relatively small values of λ (≈ 0.1) suffice for large energy savings (>80%).

All of the images displaying the results clearly show a rapid increase in energy savings for even small acceptable loss values. This increase is caused by the light sensor, which consumes an order of magnitude more energy than the vibration sensor for example (see Tab. 1). The light sensor is the first to be shut off, creating the steep climb over low values of λ seen clearly in Figs. 4 and 5. Another slight increase can be seen around $\lambda = 0.5$ corresponding to the acceleration sensor. Shutting off this sensor however, causes large increases in recognition loss. In other words, the algorithm filters out those sensors first which contribute little, but cost a lot.

4.3 Classifier Comparison

The kNN classifier performance for the reference classifier (all sensors on) remained stable across κ with recognition rates between 79.6 - 80.1%. Reference recognition rates for the HMM on the other hand varied in performance from 70.5% for $\kappa = 0.125$ to 81.6% for $\kappa = 0.875$, indicating that the recognition rates of the HMM are quite dependent on the predictability of the scenario. This can be seen again in Fig. 6, where recognition losses vary little for all values of κ for the kNN classifier, but are further spread out for the HMM classifier. However, the recognition loss for the HMM is consistently higher than for the kNN classifier for the same parameters. This can be seen when comparing the top left and bottom left images in Figs. 4 and 5.

On the other hand, the kNN classifier appears to be consistently better at conserving energy than the HMM classifier, as seen in Fig. 7 when comparing energy savings of the HMM and kNN classifiers for $\lambda = 0.1$ or $\lambda = 0.6$ for example. Fig. 5 demonstrates that this is also evident for other values of κ. Both Fig. 4 and Fig. 5 indicate that the energy consumption of the kNN classifier is also less for higher values of ρ, staying constant where the HMM energy savings fall off. Fig. 8 confirms this (noisily) by indicating higher savings for the kNN classifier compared to the HMM, and less variance over κ for higher values of ρ.

5 Discussion and Insight

In Figs. 4 and 5, non-zero energy savings are present, even when $\lambda = 0$. Intuitively, setting $\lambda = 0$ means that any loss in recognition is unacceptable. For certain classes, some sensors are so insignificant that shutting them off results in an error increase so small that it is approximated to 0. Here the expensive light sensor is useless for most

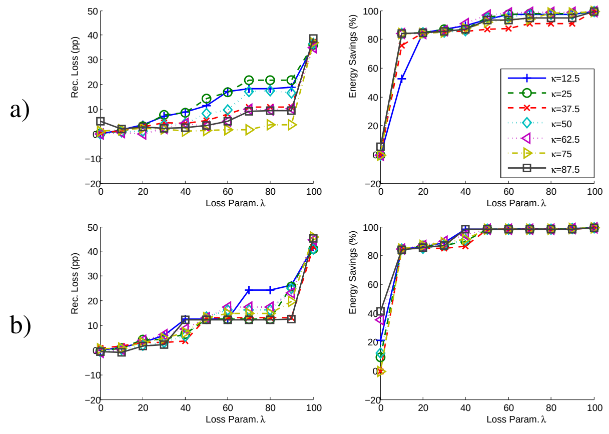

Figure 6. Recognition Loss and Energy Savings for the HMM (a) and kNN (b), ρ=4

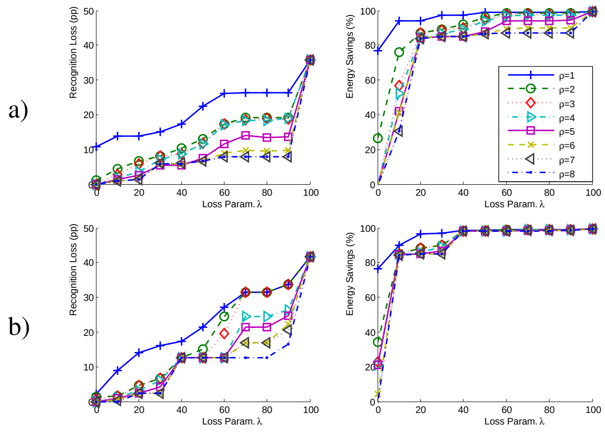

Figure 7. Recognition Loss and Energy Savings for the HMM (a) and kNN (b), κ=0.125

classes, and can be shut off with no loss as long as one of the few classes requiring it is not predicted. As more states are predicted however, classes requiring that sensor are more frequently predicted, regardless of their occurrence rate, increasing consumption with no effect on recognition.

The situation when $\rho = 1$ is extremely volatile, since only the single most likely future class is predicted. Fig. 4 and Fig. 5 show that $\rho = 1$ has significant negative effects for all non-zero values of λ. False classification results in false prediction, results in false classification again. This snowball effect is due to the fact that the system does not know when it has made an error. Introducing a confidence value at this point may allow the system to recognize error occurrence and correct by switching sensors back on. The low recognition rates for $\rho = 1$ indicate that for all real scenarios $\rho = 1$ should not be considered. For higher values of κ such as 0.875 in Fig. 5, predicting as few as two states at each step can be sufficient.

The kNN classifier was more resistant to noise with respect to predictability within the data set. As κ decreases, Fig. 6 indicates that for a fixed loss parameter λ, the recognition loss expands faster for the HMM than for the kNN. In Fig. 8, the energy consumption for a fixed ρ grows faster and becomes more erratic for the HMM when compared to the kNN. The HMM algorithm models the activities as a Markov process [11], meaning that unpredictable feature vectors not only affect prediction, but classification as well. The kNN algorithm is only affected by κ through incorrect sensor activations which reduce recognition.

Both classification algorithms are influenced by lower values of κ due to sub-optimal sensor activations. The effect can be counteracted by increasing the number of classes predicted ρ, improving recognition accuracy but reducing the gain in energy. The parameter ρ controls the balance between risk and reward. High values of ρ mean less risk but a smaller payoff, and lower values increase the win in energy at the cost of recognition. The predictability of a scenario can be easily obtained for real scenarios by taking the accuracy of the prediction algorithm over the training data. Once κ is known, ρ can be configured to counteract it and select an appropriate risk level using Fig. 8 as a heuristic.

Once the risk and reward trade off between ρ and κ has been found, the loss parameter λ can be assigned to optimize the amount by which recognition may be reduced, and thereby the amount of energy which is conserved. For example, assuming a κ value of 0.5, $\rho = 3$ to counteract and a loss parameter of $\lambda = 0.2$, a HMM incurs a loss of less than 1.2 pp in recognition but saves up to 84.11% of energy consumed without optimization. One caveat is that due to the nature of prediction, the system may perform badly for recognition of important but rare and unpredictable events.

6 Conclusion

This work proposes a novel method for recognizing human activities using embedded and wearable sensing systems. Human beings are repetitive and periodic creatures, therefore what we do can be predicted to a certain extent. Sensors which are not needed to decipher probable activities from each other can be turned off, conserving energy without greatly impacting recognition rates.

The algorithms are simulated using a preexisting data set [6], which is used to generate artificial scenarios with specific degrees of predictability. The results indicate that as the predictability of activities decreases, the wrong sensors are switched on and off, resulting in classification errors. This effect can however be counteracted by adjusting the number of activities which are predicted at each step, bal-

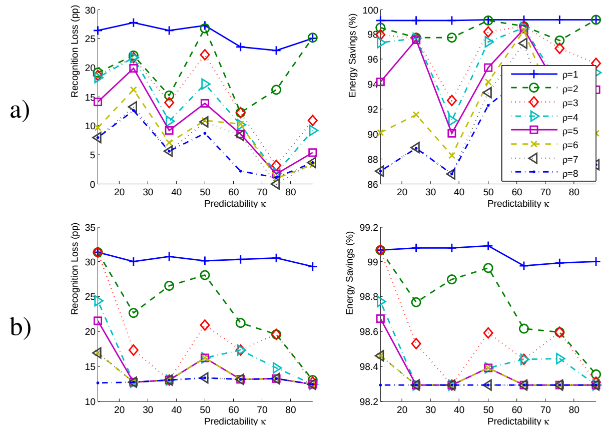

Figure 8. Recognition Loss and Energy Savings for the HMM (a) and kNN (b), λ=8

ancing risk (loss in recognition) and reward (energy savings). Finally, the application's acceptable loss parameter selects how much recognition can be traded for energy. The results indicate significant energy savings at low cost. For a scenario with predictability of 0.5 and 3 classes predicted, a loss parameter of 0.2 would incur a recognition loss of less than 1.2 pp but save up to 84.11% of energy consumed.

7 The Next Steps

These results are being confirmed and extended using other activity recognition data sets. Integrating a classification reliability measure would provide a heuristic for estimating incorrect prediction/classification combinations online in real time, and could potentially improve recognition rates. The indirect effects of missing data on prediction (via classification) should be investigated. Furthermore, this approach shows great promise for opportunistic energy savings when certain sensors are already in use by applications.

References

[1] A. Y. Benbasat and J. A. Paradiso. A framework for the automated generation of power-efficient classifiers for embedded sensor nodes. In *SenSys '07: Proceedings of the 5th intl. conf. on Embedded networked sensor systems*, pages 219–232, New York, NY, USA, 2007. ACM.

[2] N. B. Bharatula, M. Stäger, P. Lukowicz, and G. Tröster. Empirical Study of Design Choices in Multi-Sensor Context Recognition Systems. In *IFAWC: 2nd International Forum on Applied Wearable Computing*, pages 79–93, Mar. 2005.

[3] C. M. Bishop. *Pattern recognition and machine learning*. Springer, 1st ed. 2006. corr. 2nd printing edition, Oct. 2006.

[4] C. Bouten, K. Koekkoek, M. Verduin, R. Kodde, and J. Janssen. A triaxial accelerometer and portable data processing unit for the assessment of daily physical activity. *Biomedical Engineering, IEEE Trans. on*, 1997.

[5] O. Cakmakci, J. Coutaz, K. V. Laerhoven, and H. werner Gellersen. Context awareness in systems with limited resources. In *In Proc. of the third workshop on Artificial Intelligence in Mobile Systems (AIMS),*, pages 21–29, 2002.

[6] D. Gordon, H. Schmidtke, M. Beigl, and G. von Zengen. A novel micro-vibration sensor for activity recognition: Potential and limitations. In *Wearable Computers (ISWC), 2010 International Symposium on*, pages 1 –8, oct. 2010.

[7] D. Gordon, S. Sigg, Y. Ding, and M. Beigl. Using prediction to conserve energy in recognition on mobile devices. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, pages 364 –367, march 2011.

[8] V. Könönen, J. Mäntyjärvi, H. Similä, J. Pärkkä, and M. Ermes. Automatic feature selection for context recognition in mobile devices. *Pervasive and Mobile Computing*, 2010.

[9] A. Krause, M. Ihmig, E. Rankin, D. Leong, S. Gupta, D. Siewiorek, A. Smailagic, M. Deisher, and U. Sengupta. Trading off prediction accuracy and power consumption for context-aware wearable computing. In *Wearable Computers, 2005. Proceedings. Ninth IEEE International Symposium on*, pages 20–26, Oct. 2005.

[10] R. Mayrhofer, H. Radi, and A. Ferscha. Recognizing and predicting context by learning from user behavior. In W. S. G. Kotsis, A. Ferscha and K. Ibrahim, editors, *Proc. MoMM 2003: 1st International Conference On Advances in Mobile Multimedia*, volume 171, pages 25–35. Austrian Computer Society (OCG), September 2003.

[11] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257 –286, feb. 1989.

[12] N. Roy, A. Misra, and S. K. Das. Efficient long-term quality-of-inference (qoinf)-aware context determination in pervasive care environments. In *Proceedings of the 2nd International Workshop on Systems and Networking Support for Health Care and Assisted Living Environments*, HealthNet '08, pages 10:1–10:3, New York, NY, USA, 2008. ACM.

[13] S. Sigg, D. Gordon, G. Zengen, M. Beigl, S. Haseloff, and K. David. Investigation of context prediction accuracy for different context abstraction levels. *Mobile Computing, IEEE Transactions on*, PP(99):1, 2011.

[14] M. Stäger, P. Lukowicz, and G. Tröster. Implementation and evaluation of a low-power sound-based user activity recognition system. In *ISWC '04: Proceedings of the Eighth International Symposium on Wearable Computers*, pages 138–141, Washington, DC, USA, 2004. IEEE Computer Society.

[15] M. Stäger, P. Lukowicz, and G. Tröster. Power and accuracy trade-offs in sound-based context recognition systems. *Pervasive and Mobile Computing*, 3:300 – 327, 2007.

[16] F.-T. Sun, C. Kuo, and M. Griss. Pear: Power efficiency through activity recognition (for ecg-based sensing). In *Pervasive Computing Technologies for Healthcare (PervasiveHealth), 2011 5th Int. Conf. on*, pages 115 –122, may 2011.

[17] K. Van Laerhoven, D. Kilian, and B. Schiele. Using rhythm awareness in long-term activity recognition. In *Proceedings of the 12th Int. Symp. on Wearable Computers (ISWC 2008)*, pages 63–68, Pittsburgh, PA, USA, 2008. IEEE Press.