Using Prediction to Conserve Energy in Recognition on Mobile Devices

Dawud Gordon, Stephan Sigg, Yong Ding, Michael Beigl
Karlsruhe Institute of Technology, TecO
Karlsruhe, Germany
Email: [firstname.lastname]@kit.edu

***Abstract*—As devices are expected to be aware of their environment, the challenge becomes how to accommodate these abilities with the power constraints which plague modern mobile devices. We present a framework for an embedded approach to context recognition which reduces power consumption. This is accomplished by identifying class-sensor dependencies, and using prediction methods to identify likely future classes, thereby identifying sensors which can be temporarily turned off. Different methods for prediction, as well as integration with several classifiers is analyzed and the methods are evaluated in terms of computational load and loss in quality of context. The results indicate that the amount of energy which can be saved is dependent on two variables (the acceptable loss in quality of recognition, and the number of most likely classes which should be accounted for), and two scenario-dependent properties (predictability of the context sequences and size of the context-sensor dependency sets).**

***Keywords*-context recognition; context prediction, machine learning; embedded and mobile systems;**

I. 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 [1] as processing power and memory improve over time according to Moore's Law. Unfortunately, energy storage and consumption on such devices are not subject to the same doubling effects and are 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 pervasive computing applications to be practical, methods for low power situational recognition must be embedded in mobile devices.

Embedded classification for mobile devices is not a new concept and goes as far back as 1997 [2], 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 community [3][4][5][6], and trade-offs that must be made between classification quality and energy consumption for embedded context recognition are discussed in [7], [8].

While these approaches effectively reduce power consumption in certain situations, we propose a method for further reduction based on the concept of context prediction. Context prediction as studied in [9] can be used to make a prediction about future situations based on situations recognized in the past. By developing a dependence mapping between contexts, features and sensors, we propose to use context prediction to parameterize sensor usage, sampling and feature generation in order to further reduce unnecessary power consumption while minimizing the negative effect on classification accuracy.

II. Context, Feature and Sensor Mappings

The standard process for situational recognition using machine learning algorithms is straightforward. Sensors are sampled in parallel at an arbitrary rate for an arbitrary period of time, after which the data is then saved as a discrete multidimensional array, referred to as a sample window. This window is processed using different algorithms to generate so called features, e.g. standard deviation, average, FFT or cepstral coefficients. Which features are used depends on the application, i.e. which situations we want to recognize and the type of sensor being used and are referred to all together as a feature vector. The feature vector is then passed to a machine learning algorithm whose task is to recognize which situation was occurring during the sample window, based on its feature vector.

When observing this chain of events in the context classification process, it should be clear to the reader that each feature in the set of features used $f \in F$ is implicitly mapped onto a single sensor in the set of sensors $s \in S$, namely the single sensor which generates the data for this feature, producing the surjective mapping a of features onto sensors: $a\colon F \to S$

Mapping recognition classes onto the features is not as simple and requires a bit more legwork. The concept of selecting features which best suite an application is not new, Könönen et al. provide an overview of feature selection algorithms for embedded systems in [10]. While these algorithms potentially improve the quality of classification and reduce

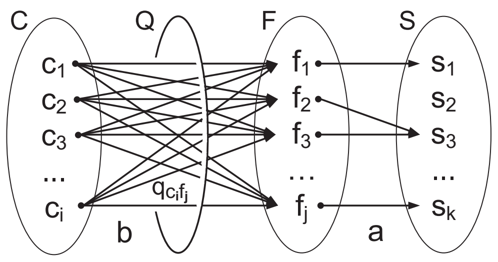

Figure 1. (C)lass, (F)eature, (S)ensor Mappings (a,b) and Weights (Q)

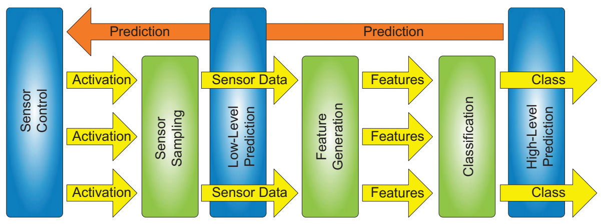

Figure 2. Integration of the Classification and Prediction Processes

the computational load, they do not provide a mapping of features to classes by relevance or importance.

It should be clear to the reader that turning sensors on and off will result in a dynamic feature 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. Hidden Markov Models are also well suited as the observational distributions for these variables are ignored when calculating the probabilities of the hidden states. Both of these examples lose only the information that would have been gained from the missing features, but are not further negatively affected [11].

In order to generate the weighted mapping, training data is gathered for each class. After training the classifier over all of the training data, each class is tested for dependency against each feature. This is done by testing the trained classifier against all of the training vectors for each class and removing each feature one at a time (all other features are reinstated), and the degree of dependency is inferred using the the drop in accuracy when a feature is removed: a large drop in recognition indicates a high dependency, a small drop, low dependency.

The result of this is a weighted mapping $b\colon C \xrightarrow{Q} F$ of classes C onto features F with quality values $q \in Q$ where q has a value from 0 to 1, namely the cost of that feature for that class in percent loss in recognition accuracy. Both mappings can be seen in Fig. 1, where Each class $c \in C$ is mapped onto each feature $f \in F$ over the parameterized mapping b with a quality weight $q_{c_i f_j} \in Q$ for class i and feature j, and each feature is in turn mapped to one sensor $s_k \in S$ over the mapping a.

As each feature will incur a certain loss in recognition $q \geq 0$, and it is assumed that at least one q will be non-zero for each class, it is necessary to decide what loss is acceptable for each application: l. Using the mappings a and b and weights Q we must now calculate the total cost ω of each sensor with respect to each class, as this is required in order to judge if turning off a sensor will violate l.

For each sensor s_j the subset $F_{s_j} \subseteq F$ is built which represents all features generated using that sensor. For each class c_i, the subset $Q_{c_i s_j} \subseteq Q$ is then built, which contains the costs q of all features $f \in F_{s_j}$, or the cost of each feature generated using sensor s_j with respect to class c_i. Now the total cost of a sensor with respect to a class can be calculated $\omega_{c_i s_j} = \sum_{q \in Q_{c_i s_j}} q$. If Ω_{c_i} is the set of ω for all sensors with respect to the class c_i, then in order to perform the sensor optimization for class c_i, the sensor $s_n | \forall j; \omega_{c_i s_j} \leq \omega_{c_i s_n}$, meaning the sensor ($s_j$) of least importance to that class (c_i), can be turned off and $\omega_{c_i s_j}$ subtracted from l. This is repeated until l is depleted, meaning until removing the next least pertinent sensor would violate the acceptable loss threshold for the specified application.

Now, for each class c_i, a set of sensors S_{c_i} has been identified which is required in order to recognize that class, and more importantly, we can identify sensors which are not of interest given the acceptable accuracy loss l. 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, and will allow us to shut off sensors which are not needed to conserve energy.

III. Context Prediction

The general approach to using context prediction to conserve energy is simple: the system should be able to predict a subset of classes which are likely to occur in the near future, i.e. the next sample window. Using this information and the mapping of classes onto features onto sensors, it is logical that the set of sensors necessary to recognize that class can be identified and activated as shown in Fig. 2, providing an energy advantage over systems which use all sensors at all times, while maintaining loss of accuracy below l.

Since we use prediction to deactivate sensors, we also have to consider the impact of the absence of this sensor data on prediction accuracy. This means that the input time series dimension is dynamic, which might reduce the prediction accuracy as the information provided is reduced. This affect is especially serious when prediction is carried out before feature data is aggregated or classified (low-level prediction). In [12] we derived that prediction accuracy is affected by the amount of pre-processing applied to the input time series, though in this case, this dependency must be further explored to account for fluctuation in the dimension of the input time series.

Unlike low-level prediction, a high-level approach (using feature or context data) is typically applied to an input time series with fewer dimensions and a reduced sample space. Since the time series of feature values are already aggregated and classified at high-level prediction, it is an inherently easier task when compared to low-level prediction, and the effects of the of the dynamic sensor configuration are reduced. Several algorithms such as ARMA [13] which can be applied to numerical time series data, can not be applied to the symbolic context data which is likely to be found in high-level context time series. A very general method for context prediction on high-level data is to utilize a Markov process [14], which is optimal in the sense that it can always achieve the highest possible prediction accuracy for infinite binary random sequences [15].

Using a Markov Chain trained during classifier training for illustration, we could use this chain to predict the set of states, or classes, which are likely to occur in the next sample. As with the acceptable recognition loss, there is once again an application-based trade-off which has to be made: where to draw the line between what is a likely enough state to be accounted for in the near future, meaning that the sensors required for that class should be activated before entering the next sampling phase. This decision can be made by selecting set of the p most likely classes, where $1 \leq p \leq N$ for a scenario with N classes.

IV. Evaluation

Computational Load: To generate the class to feature parameterized mappings, each iteration requires only one classification step using the trained classifier, yielding $\mathcal{O}(NM)$, where N is the number of classes and M is the number of features. This is only possible if a classifier is used which can handle a variable feature vector length. If this is not the case, a new classifier must be trained for each different variable combination in order to compare classification rates for the class-feature mapping weights which would increase computational load by the cost of classifier training. This would not only greatly increase computational load at training time, but memory usage at runtime as $N \times M$ classifiers must be stored locally for classification.

The task of prediction must be periodically carried out on the node, which is why analyses of these processes is also of interest for this work. A comparison between several prediction approaches regarding their computational load is not always straightforward, since the load is not always directly comparable. Consider, for instance, the ARMA and the Markov prediction methods. When the length of the observed context time series is k and $\mathcal{C}$ denotes the number of possible context values, context prediction using ARMA methods has an asymptotic complexity of $\mathcal{O}\left(k \log(k)\right)$, plus the cost of classifying the predicted data. For Markov prediction methods, the complexity is $\mathcal{O}\left(|\mathcal{C}|\right)$ [9].

For high-level prediction, the prediction step itself is computationally very simple and can easily be optimized. A Markov Chain can be represented by an $N \times N$ matrix, where N is the number of classes between which the system should distinguish. After each classification, the probability for each class can be retrieved from the table, and the p highest probabilities are selected. This computation is a linear search through one row or column of the matrix, which grows as $\mathcal{O}(N)$ where N is once again the number of classes which we are trying to distinguish. Once p has been fixed, the p most likely next states for each state become static and the operation becomes $\mathcal{O}(1)$.

Dependencies: In Sec. III, two variables were identified which would allow the application designer to affect the energy/accuracy trade off. The first is l, or the acceptable recognition loss for the specific application or scenario. Setting this value closer to 1 creates a system with possibly low recognition rates, but higher energy savings, while for a value closer to 0 the behavior approaches that of a system without any energy optimization. The second variable which can affect the trade off is p, or the set of states which should be accounted for in the next sample window. Following Sec. III, as this value approaches 1, the energy savings are maximized, but so too is the possible loss in recognition rates, as the set of recognizable classes is reduced to one. As p approaches N the power savings and recognition loss rates approach that of a system without optimization, e.g. 0.

Furthermore, two properties intrinsic to the particular scenario were also identified which will affect the energy/accuracy trade off in the system. The first the predictability of the data, which is a function of the scenario itself, as well as the prediction method being used [13], [14]. Unpredictable data will lead to large errors in the prediction causing power savings to come at disproportionately high recognition rate costs. On the other hand, if predictions are accurate, power savings would be far more affordable. The second property is the grouping of the sensor/feature dependencies, or the size of $S - S_{c_i}$ for each class, and the costs ω of those sensors. To demonstrate, imagine a system in which each class is equally dependent on each sensor, meaning the sensor weights in Ω are equally distributed for each class. Turning off any sensor could then lead to a direct violation of the acceptable loss l, forcing the system to maintain all sensors on at all times. In the opposite case, imagine a system where each class is dependent to 100% on only one sensor, meaning Ω contains one 1 and otherwise 0's for each class. Assuming flawless prediction, only one sensor would be on at any given time, optimizing energy savings with no loss in classification accuracy.

V. The Next Steps

Although the main concepts have been presented here, a full evaluation of the system is yet to be completed. This will clarify the advantages and disadvantages of high-level vs. low-level prediction methods as well as the dependencies between energy savings, recognition accuracy loss and the two system variables. The approach is two-fold, where initially the system will be tested using simulation, followed by an evaluation using a high-modality sensor board to validate the simulation results under real conditions.

One other aspect which is being explored is the further parameterization of the mappings a and b to reflect the cost of each feature and sensor in terms of power consumption, and to introduce this as a metric for further optimizing sensor and feature selection, i.e. the system will be reticent to use expensive sensors and features. Furthermore, until now the recognition algorithm has been considered as the only application running on the device with full control over sensors. In reality, especially when considering smart phones, other applications will run along side, or on top of the recognition and have their own requirements for sensing. The mappings can then be used to opportunistically improve activity recognition by evaluating feature computational costs when sensors are in use by other applications.

VI. Conclusion

In this paper we have proposed a method for conserving energy in context recognition by using prediction algorithms. A method for mapping classes onto features and weighting these according to incurred loss of recognition accuracy was put forward. It was then shown how these mappings allow predictions to be applied to the system to turn off unneeded sensors. Two scenario-based properties which will affect the success of the approach were identified, namely the inherent predictability of the data, as well as the dependency distributions of classes over the sensors. Furthermore two system variables where introduced which will affect the ratio of power consumption and recognition rates. Specifically, these are the amount of loss in recognition rates which are acceptable, as well as how many classes should be accounted for based on each prediction. Finally, the next steps for evaluating the effects of the methods and the correlations between the crucial variables, prediction accuracy, classification accuracy and energy savings was detailed.

Acknowledgments

The authors would like to acknowledge joint funding by the European Commission under the ICT project "CHOSeN" (Project No. 224327, FP7-ICT-2007-2) and by the Deutsche Forschungsgemeinschaft (DFG) under the project "Sense-Cast " (BE4319/1).

References

[1] M. Berchtold, M. Budde, D. Gordon, H. Schmidtke, and M. Beigl, "ActiServ: Activity Recognition Service for Mobile Phones," in *ISWC'10: Proceedings of the Fourteenth International Symposium on Wearable Computers*. Seoul, S. Korea: IEEE Computer Society, 2010, pp. 83–90.

[2] 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 Transactions on*, vol. 44, no. 3, pp. 136–147, March 1997.

[3] 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), ECAI 2002*, 2002, pp. 21–29.

[4] 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*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 138–141.

[5] 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 international conference on Embedded networked sensor systems*. New York, NY, USA: ACM, 2007, pp. 219–232.

[6] 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*, Oct. 2005, pp. 20–26.

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

[8] 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*, Mar. 2005, pp. 79–93.

[9] S. Sigg, S. Haseloff, and K. David, "An alignment approach for context prediction tasks in ubicomp environments," *IEEE Pervasive Computing*, vol. Oct-Dec 2010, 2010.

[10] V. Könönen, J. Mäntyjärvi, H. Similä, J. Pärkkä, and M. Ermes, "Automatic feature selection for context recognition in mobile devices," *Pervasive and Mobile Computing*, vol. 6, no. 2, pp. 181 – 197, 2010.

[11] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2001.

[12] S. Sigg, D. Gordon, G. von Zengen, M. Beigl, S. Haseloff, and K. David, "Investigation of context prediction accuracy for different context abstraction levels," *IEEE Transactions on Mobile Computing*, 2011, (to appear).

[13] C. Chatfield, *The Analysis of Time Series: An Introduction*. Chapman and Hall, 1996, vol. 5.

[14] W. Feller, *An Introduction to Probability Theory and its Applications*. Wiley, 1968.

[15] M. Feder, N. Merhav, and M. Gutman, "Universal prediction of individual sequences," *IEEE Transactions on Information Theory*, vol. 38, no. 4, pp. 1258–1270, July 1992.