Cost-Sensitive Trees for Energy-Efficient Context Recognition

Vito Janko
Jožef Stefan International Postgraduate School
Jožef Stefan Institute, Ljubljana, Slovenia
vito.janko@ijs.si

Mitja Luštrek
Jožef Stefan International Postgraduate School
Jožef Stefan Institute, Ljubljana, Slovenia
mitja.lustrek@ijs.si

***Abstract*—An important problem in the field of context recognition is preserving the battery life of the sensing device. In this work we introduce cost-sensitive learning that takes into account the cost of having the sensors active and tries to balance it against the classification error. The benefits can be amplified by adapting the classifier to each context. This approach was tested on two real-life datasets where we achieved 78% and 80% reduction in energy consumption in exchange for almost no accuracy loss.**

***Index Terms*—context recognition, cost-sensitive learning, decision tree, Markov chains, multi-objective optimization, battery**

I. Introduction

Context recognition is a term encompassing a wide array of tasks where sensors are used for monitoring a user and trying to infer something about their context. Recognizing their activities or location using the smartphone they carry is a typical example of the task.

A common – yet not often addressed – problem in most context recognition tasks is the battery consumption of the sensing (and processing) device. For example, having all sensors in a smartphone active at all times can be helpful in a variety of context recognition tasks, but the smartphone's battery would drain quickly, making the application useless in practice. Carefully selecting which sensors are active in each predefined circumstance can mitigate the problem, but this selection can be time consuming and can require a high degree of expert knowledge. Having an automatic procedure for the selection would therefore be beneficial.

Our proposed solution uses cost-sensitive learning, which is a machine learning field concerned with both the misclassification cost and with the cost of attributes that were used to generate the classification. A typical example comes from the field of medicine, where each test on a patient has an associated cost (in money, time, discomfort, etc.), but at the same time increases the probability of a correct diagnosis. The goal is to balance the attribute costs with the misclassification cost, for example by trying to minimize their sum.

In the following sections we present a method that can – given a collected context-recognition dataset – automatically infer when and which sensors should be active. It presents different trade-offs between classification accuracy and the sensing device's energy consumption. We first explain how to apply the cost-sensitive methodology to the field of context recognition, using a modified decision tree. We then show that using different decision trees for classifying different contexts can further increase the energy savings. To do so, we use a genetic algorithm in combination with a Markov-chain model to select the appropriate tree for each context.

Different variants of this approach are tested, compared and shown to work well on two real-life datasets.

II. Methodology

The presented methods assume that we have a dataset collected from different sensor streams. Data is then split into individual instances (composed of different attributes – each sensor stream may generate one or more attribute), and the goal is to use a machine learning model to classify all of the them, each to one of predetermined contexts. We aim to reduce the expected cost of this classification – that is, the sum of the misclassification cost and the cost of the attributes used.

A. Cost-sensitive decision tree

A cost-sensitive decision tree (CS-DT) was used as the machine learning model throughout this work, due to its popularity as a cost-sensitive model and due to how naturally it can be adapted to the context recognition setting. There are many CS-DT described in literature [6]; we implemented the CS-DT as presented in the work of Ling et al. [5], and modified it as described in Section II-B. This particular implementation was chosen because it makes the sum of the attribute and misclassification cost explicit, and the trade-offs between them easy to understand. To explain it briefly: at each tree node – instead of checking each attribute's information gain or a similar metric – we compare the expected cost of misclassification if this node becomes a leaf, with the expected cost of the attribute and misclassification if the attribute is used to further divide the instances. The attribute that reduces the expected cost the most, if any, then expands this node.

An immediate concern is that attribute and misclassification costs are measured in different units with no obvious way to combine them. The most straightforward way is to decide on a trade-off in advance (for example, one might be willing to sacrifice 2% of accuracy for an extra hour of battery life). In this work we attempt to bypass this problem by presenting multiple trade-offs to the user. These trade-offs (represented by CS-DTs) form a Pareto front – so no trade-off is better in both accuracy and energy consumption than another. System designer can then choose one that fits the application needs.

2472-7571/19/$31.00 ©2019 IEEE
DOI 10.1109/IE.2019.00011

B. Using CS-DT for context recognition

In the case of context recognition, each attribute is calculated from data that is coming from one of the sensors. Its cost comes from having that sensor active, draining the battery of the sensing device (with the additional battery drain resulting from the data processing being implicitly included in this cost). This context recognition setting is somewhat different from the standard cost-sensitive setting; the differences and our adaptations are listed below:

1) Delayed sensor activation: In a real-life setting it is impossible to retrieve a sensor reading for the current instance, if the sensor is not currently active. Therefore, we cannot simply traverse the decision tree and calculate the attributes as necessary. We can, however, leverage the fact that the next instance will be similar to the current one, and that probably the same sensors will be needed. The CS-DT is modified so that while it is classifying an instance, it traverses the tree until it comes to a leaf node or a node with an unknown attribute. Then, it classifies that instance using that node as a leaf node, and in the case of an unknown attribute, it signals to the system to turn the needed sensor on. It also notes which sensors were not needed, and they are turned off for the next instance. This way, the system gradually and fluidly adapts its sensing capabilities to the current need.

2) Shared attribute costs: Data stream from a single sensor can be transformed into many different attributes. These attributes have a shared cost – calculating the first requires activating the appropriate sensor, while the subsequent attributes derived from the same sensor are essentially free. Furthermore, the costs of different sensors are not independent from each other (e.g. look at the Commodity12 dataset in Section III-A). To solve both problems, the CS-DT was adapted so that when the tree is traversed, it tracks which attributes were used in all previous nodes. These attributes serve as an additional input for the cost function of the current node's attribute.

3) Taking advantage of shared attribute costs: Tree generation process is myopic – each attribute is evaluated individually. This problem is compounded by the shared attribute costs. One attribute, for example, may not be worth activating the corresponding sensor for, but all the attributes from that sensor could be. To partially mitigate this, we changed the CS-DT generation so that when at a particular node no attribute reduces the expected cost, the best attribute expands the node anyway. This can happen maximally n times in each tree branch (n is a parameter). This is often beneficial because the attribute selected for the next node is often free and the two can be jointly useful. The tree is pruned after being generated.

4) Advanced use of shared attribute costs: An alternative to the solution presented in Section II-B3 is to drastically change the tree-generation process. At each node, instead of choosing the best attribute, we chose the best sensor by comparing subtrees potentially containing all attributes using a single sensor. The best sub-tree replaces the node and this process is then repeated at each leaf of the sub-tree. This resulted in most cost-efficient trees, but at the cost of higher generation time.

C. Generating different CS-DT

This section lists three methods for generating different CS-DTs. These CS-DT are useful for getting different trade-offs between energy consumption and classification accuracy, and for assigning them to different contexts (Section II-D).

1) Changing misclassification cost: Different CS-DT can be generated by changing the misclassification cost. Note that these trees are all semantically similar – usually only increasingly smaller as the misclassification cost falls.

2) Sensor subsets: Using this method, one CS-DT was generated from each sensor subset – using only attributes from those sensors. The rationale is that different contexts may require different sensors to detect them. In the case of the Opportunity dataset (Section III-B), only a partial subset containing greedily determined most useful sensors was used due to the combinatorial complexity of the task.

3) Binary trees: One CS-DT was generated for each context. A context-specific tree for the context c only classifies instances into "c" or "not c". To use these trees, the system architecture is changed to perform a two step classification. In the case of "not c", a general-purpose decision tree is used to classify the next context. The whole dataset – with modified labels – is used for each context-specific tree generation. To generate more trees, we combined this approach with the one described in Section II-C1.

D. Using different CS-DT for each context

Classifying all instances using the same CS-DT can sometimes be inflexible. For example, the sensor at the root of the tree will always be active regardless of which context the user is currently in. Having a separate tree for each context and using it when that context is recognized might thus be be preferable. Intuitively, a different rule set for sensor switching should be used depending on whether the user is resting at home, for example, or running outside. This approach adapts to both the current sensor data (within CS-DT) and the last classified context (switching between the CS-DTs), allowing the system to find interesting energy-efficient solutions.

Having generated a set of CS-DTs as described in the previous section, the last task is to find a good assignment of what tree to use for each context.

1) Finding good assignments: This problem can be treated as a multi-objective optimization problem – with classification quality (accuracy for example) and energy consumption – being the two objectives. We can therefore use some of the methodology from this domain.

In this work we used the NSGA-II [3] algorithm due to its widespread use, although another similar algorithm could probably fill the same role. Different tree-to-context assignments were used as the population in the algorithm, and the Markov-chain simulation (described in Section II-D2) was used to estimate their performance. The output of the method is a Pareto front of different solutions – the best found trade-offs between classification quality and energy consumption. Having this Pareto front, the application designer can then pick a suitable solution, depending on the application needs.

2) Evaluation: Given a tree-to-context assignment, we must evaluate the overall performance of the corresponding system. It would be naive to simply do a weighted average of used CS-DT performances (for example, having three equally represented contexts, with their respective trees showing accuracies of 80%, 85% and 90% to assume the whole system will work with the accuracy of 85%). What often happens in practice is that a missclassification activates the CS-DT for a wrong context, which is ill-suited for the actual context, leading to further classification errors.

The straightforward way for performing the evaluation is to run the system through the whole dataset, simulating the sensor changes on the way. This can be time consuming, especially on larger datasets.

The used alternative is to simulate what would happen using a Markov-chain model. The details of this simulation can be found in our previous work [4]. In summary, we can use the confusion matrices of the individual trees and transition probabilities between the contexts to define the Markov-chain states. Then we can derive the so called steady-state of the chain, from which we can accurately evaluate the system.

III. Datasets

A. Commodity12 dataset

A real-life dataset, where a smartphone and a chest-worn heart-rate monitor were used to monitor 10 participants. Each participant captured continuous two weeks of data and hand-labeled the following contexts: sleep, work, home, eating, transport, exercise, out (out of house, but not in any of the previous contexts). The data came from ten different physical and virtual sensors: accelerometer, barometer, light sensor, GPS location, a list of visible WiFi networks, location description by the Foursquare web service, sound, time, heart rate and respiration rate. The first eight were measured with the smartphone, and the last two with the heart-rate monitor, which was connected to the smartphone via Bluetooth. Attributes were calculated for each minute of the data, and one minute became one learning instance. All details can be found in our previous work [2]. While the classification accuracy was reasonably high, the energy consumption of the application prevented it from seeing any practical use.

Predicting the energy consumption of a smartphone is not trivial. Due to sensors sharing hardware and various software optimizations, the cost of activating sensors does not add up linearly. For example, the cost of accelerometer and gyroscope being active is much lower than the sum of their individual costs. To create a cost function, we measured the energy consumption of all combinations of sensors on a Samsung Galaxy S4 device. We made the following simplifications: 1) the time "sensor" is free, 2) the heart rate and respiration are free as long as the Bluetooth is active, 3) the Foursquare web service is free if both GPS and Wi-Fi are active and 4) having one, two or all of the following sensors – accelerometer, light sensor and barometer – resulted in a very similar energy consumption, perhaps because they share many resources to operate. The remaining combinations are shown in Table I.

Combination	Current [mA]	Combination	Current [mA]
/	20	A	46
B	45	W	50
S	55	A, B	52
B, W	75	B, S	68
A, W	70	A, S	61
W, S	84	A, B, W	74
A, B, S	70	B, W, S	90
A, W, S	85	A, B, W, S	100

TABLE I
ENERGY COST OF DIFFERENT SENSOR COMBINATIONS. SENSORS THAT WERE ACTIVE WERE LABELED A – ACCELEROMETER, S – SOUND, B – BLUETOOTH, W – WIFI.

B. Oportunity dataset

Opportunity [1] is a popular, publicly available dataset. Four users were recorded while performing different tasks in an apartment. In that apartment, there were 30 sensor clusters, some on the user's body, and some on objects that the user interacted with. Each cluster contains some of the following sensors: accelerometer, gyroscope and magnetometer. The dataset provides different sets of labels, out of which we decided on the Motion labels, which annotate the following four activities: sit, stand, walk and lie.

For the purpose of this work, we decided to count each sensor cluster as one sensor, and each cluster was assigned the same cost. Note, however, that the same methodology could be used with a more complex energy cost function if needed.

IV. Results

Our first test was to measure the energy saved by CS-DT due to not having all the sensors active all the time. The same test also measured the accuracy loss, which is a negative consequence of the delayed sensor activation (Section II-B1). All generated CS-DT were tested with both sensor switching and with the assumption that all sensors are always active. Table II presents the results. In both cases we lost a small amount of accuracy in exchange for a substantial drop in energy consumption. The Opportunity dataset has a higher accuracy loss due to the quickly changing contexts (e.g. standing, walking) as opposed to to the Commodity dataset (e.g. work, home). The Commodity12 dataset has a proportionally smaller energy gain due to having a baseline energy cost (there is a cost even when no sensors are active).

	Commodity12	Opportunity
Accuracy loss [%]	0.2	2.2
Energy reduction [%]	27	44

TABLE II
AVERAGE DIFFERENCE BETWEEN THE ACCURACY LOSS AND ENERGY GAIN BETWEEN REGULAR AND COST-SENSITIVE DECISION TREES.

To have a better picture of the trade-offs between the accuracy and energy consumption, we generated a Pareto front of possible solutions for both datasets using different methods. The results are shown in Figure 1 and Figure 2 for Commodity12 and Opportunity datasets, respectively. In these graphs, the lower left point is the ideal one. First we plotted the trade-offs using classical decision trees that use data from different sensor subsets (Section II-C2). Then, the same process was repeated using CS-DTs. The results of this experiment

showed that CS-DT offers similar accuracy with substantially lower energy consumption. Note that these are the same trees as those used for Table II. Next, we tried to get different solutions by having different weights for the misclassification cost (Section II-C1), and left the sensor selection to happen automatically in the tree generation process. Doing so, we got comparable results. Finally, we tested systems that switch between different CS-DTs, depending on the last classified context – using both sensor subsets (Section II-C2) and binary tree (Section II-C3) methods. This resulted in best solutions, with binary trees being slightly worse due to the somewhat energy-inefficient two-step classification.

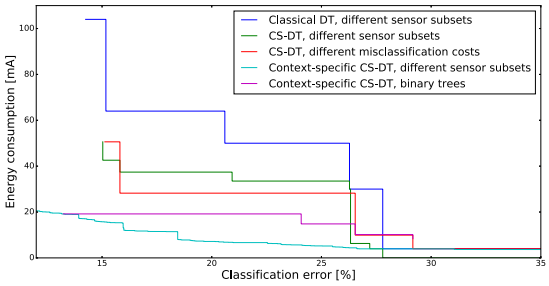

Fig. 1. Different trade-offs for the Commodity12 dataset.

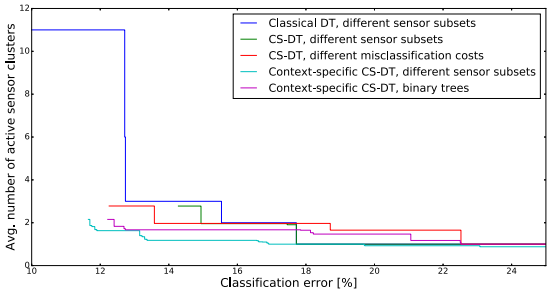

Fig. 2. Different trade-offs for the Opportunity dataset.

We found the generated solutions semantically sensible. For example, a small part of one context-specific solution for Commodity12 is presented in Figure 3. One can observe that a completely different sensor subset can be used when classifying, depending on the last classified context.

Finally, we looked at different ways for constructing a CS-DT. In the experiments done on the Commodity12 dataset (Figure 4), both extending the tree generation and grouping the attributes together helped to reduce the problem of attributes sharing cost. The former approach ($n = 1$) was used, as with it the generation was roughly 5 times faster than with the latter.

V. Conclusion

In this work we proposed a method that can reduce the energy consumption of a context-recognition system. It does so by automatically (using no expert knowledge) determining which specific sensors are needed for the task and proposing rules for when to switch between them. The method combines the idea from our previous work [4] – to use different sensors when different contexts are detected – with cost-sensitive

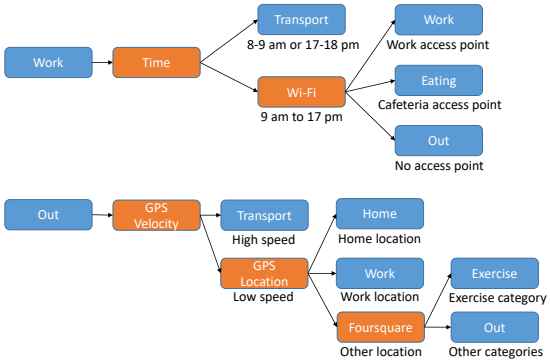

Fig. 3. Parts of the CS-DTs assigned to the contexts "work" and "out". Orange nodes show which sensor must be activated. Blue nodes represent the classification.

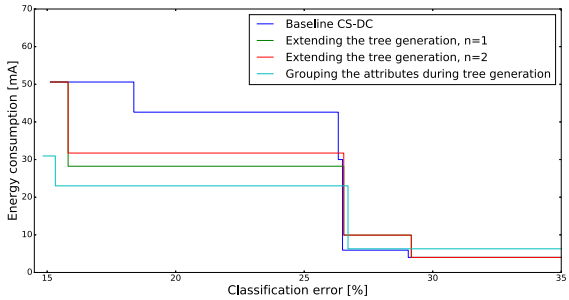

Fig. 4. Different trade-offs (using different misclassification weights) for different tree construction methods on the Commodity12 dataset.

learning. The novelty of the method is not only in combining the two ideas, but also in adapting the cost-sensitive learning to the problem of context recognition.

The use of cost-sensitive decision trees drastically improved the energy consumption, losing only few percentage points of accuracy in the process. The effect was compounded when each context used a separate decision tree. Looking at the solutions with the highest accuracy for each method, we see that the context-specific trade-offs are 78% and 80% more energy-efficient than the classical decision tree.

References

[1] R. Chavarriaga, H. Sagha, A. Calatroni, S. T. Digumarti, G. Tröster, J. d. R. Millán, and D. Roggen. The opportunity challenge: A benchmark database for on-body sensor-based activity recognition. *Pattern Recognition Letters*, 34(15):2033–2042, 2013.

[2] B. Cvetković, V. Janko, A. E. Romero, Ö. Kafalı, K. Stathis, and M. Luštrek. Activity recognition for diabetic patients using a smartphone. *Journal of medical systems*, 40(12):256, 2016.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[4] V. Janko and M. Luštrek. Using markov chains and multi-objective optimization for energy-efficient context recognition. *Sensors*, 18(1):80, 2018.

[5] C. X. Ling, Q. Yang, J. Wang, and S. Zhang. Decision trees with minimal costs. In *Proceedings of the twenty-first international conference on Machine learning*, page 69. ACM, 2004.

[6] S. Lomax and S. Vadera. A survey of cost-sensitive decision tree induction algorithms. *ACM Computing Surveys (CSUR)*, 45(2):16, 2013.