Conditional Random Field Feature Generation of Smart Home Sensor Data using Random Forests

Eastwood, M., Konios, A., Tan, B., Jing, Y. & Hamid, A.

DOI 10.1109/IMBIOC.2019.8777764
ISBN 978-1-5386-7395-9

Publisher: IEEE

Conditional Random Field Feature Generation of Smart Home Sensor Data using Random Forests

M. Eastwood*, A. Konios*, B. Tan*, Y. Jing† and A. Hamid‡
* Institute for Future Transport and Cities, Coventry University, United Kingdom.
† Faculty of Business and Law, Coventry University, United Kingdom.
‡Faculty of Engineering, Environment and Computing, Coventry University, United Kingdom.

***Abstract*—A typical approach to building a feature set for a conditional random field model is to build a large set of conjunctions of atomic tests, all of which adhere to a small number of relatively simple templates. Building more complex features in this way can be difficult, as the more complex templates needed to do this can result in a combinatoric explosion in the number of features. We use the inherent instability of decision trees to produce a small set of more complex conjunctions that are particularly suitable for the problem to be solved, using the same techniques used in generating random forest ensemble classifiers, and build a CRF on these features. We apply this method to an activity recognition problem on a dataset from the CASAS smart home project, in which we predict activities of daily living from sensor activations.**

I. Introduction

With the increase in availability and sophistication of wireless sensor networks allowing a constant stream of ambient data to be collected in smart homes, activity recognition (AR) has become an actively researched field. From data provided by sensors such as motion, power usage, light, temperature and door sensors, it is possible to recognise many everyday activities a resident is engaged in, as well as recognising particularly abnormal behaviour [1]. This has applications in many areas (e.g. healthcare, security, etc.) and could potentially become extremely important in elderly care.

As people get older and frailer, many want help to be quickly available if needed, but also want to maintain as much independence as possible. Further, relatives want to know that there is a safe place where their loved one is okay without constantly checking up on them. Smart homes paired with activity recognition and data analytics can provide this security while preserving independence. A system that can provide a well-being indicator, and sound an alarm in case of an accident or health problem, relies on AR to provide this functionality [2]. Recognition of everyday activities carried out as normal like making lunch, personal grooming etc. allows an indication that the resident is alright. Also, recognising hazardous or abnormal behaviour in daily activities (e.g. falls) can indicate the presence of a problem and trigger an alert for a resident. Thus, such systems require an effective AR model to function properly.

This work considers AR as a sequential prediction problem. We assume a sequence of observations X each labelled with the action that was being performed Y, where each x_t belongs to a set of possible sensor readings $\mathcal{X}$, and each label y_t to a set of possible labels $\mathcal{Y}$. The task is to predict the correct labelling of a new observation sequence.

Conditional Random Field (CRF) is a method which has been shown to perform well on a variety of sequential prediction tasks of this form [3], however the performance of the method relies heavily on the features used to build the model. Often it is not clear how to best include features into the model which can capture more complex dependencies, as the typical approach to generating features involves creating all features following a certain pattern, or template. To generate more complex features, we must define more complex templates, which can result in a combinatorial explosion of the number of features generated, making learning difficult. Here, to combat this problem, an approach is introduced using features extracted from decision trees to provide a small set of complex features particularly tailored to the prediction problem to be solved. This is applied to an AR dataset in a smart home setting.

The paper is structured as follows. The next Section introduces some background and related work. In Section III, we describe the CRF model, followed by a discussion of our approach to feature generation for CRFs in Section IV. Subsections will motivate the usefulness of features extracted from a random forest ensemble of decision trees and describe the method used to do this. Experimental work will be covered in Section V, and Section VI concludes.

II. Related Work

The dataset used to test the method proposed in this paper is an activity recognition task using data from a smart home environment. Thus, we briefly provide some background relating to smart homes and the Ambient assisted living (AAL) paradigm.

A smart home is an equipped dwelling where a variety of ambient sensors monitor the state of the living space and its occupants, and may potentially be used to control aspects of the environment. Using such information together with communications technologies and connected devices to improve quality of life, is the core of the AAL paradigm [4], with AR being an important part of this approach. Smart home setups have become steadily more sophisticated in recent years, as sensors become smaller and more widely available, and new technologies such as Bluetooth low energy, Zigbee and other wireless networking standards appear. Smart home environments have been described in the literature which use a diverse array of different sensor setups, and from which interesting datasets have been collected. Some of these projects are briefly described here.

In the SPHERE project [5], a multi-modal sensor infrastructure that can be deployed in a home environment is proposed. This consists of a video monitoring component, a wrist-worn body motion sensor, and a set of environmental sensors such as door, motion and power usage sensors. Sensors communicate with component hubs which connect to a home gateway for transferral of data to remote servers.

The CASAS smart home [6] is a lightweight non-intrusive suite of sensors designed to be quickly and easily set up. It consists of motion, door, light, temperature and water/power usage sensors. A visualisation system is provided and an activity recognition system trained to identify activities of daily living (ADL). A number of datasets, some annotated, ranging from single-occupancy apartments to medium (20 persons) workplaces are available online [7]. In this work, we use one of these, which is further described in Section V.

Patterns of normal behaviour and ADLs can be learnt from the data collected from smart homes like those described above, and used for example to provide well-being indicators, potentially identify developing health problems and critical events such as falls, or perhaps to adjust the living environment in response to some detected situation or event. The sensor data resulting from such smart home setups is best represented as a sequence of discrete events. Each event comprises a sensor ID identifying the activated sensor, a timestamp and, in the case of non-binary sensors, a value denoting a newly polled sensor output. The exception to this is data from wearables, providing a stream of axial accelerations. This can be translated to an event sequence by a processing stage identifying particular motion patterns of interest. The Activity Recognition task essentially becomes a sequence labelling task, assigning an activity label to each sensor activation in a sequence of sensor readings. Some popular sequence labelling algorithms are reviewed below.

Hidden Markov Models (HMMs) have commonly been used for sequential modelling. The system is assumed to be a Markov process which transitions between a number of hidden states which correspond to the labels of the sequence. Each state is associated with a distribution over the output observations allowing inference of the probability of the state sequence given a sequence of observations. Many variants of this method exist, such as the hierarchical HMM described in [8], which is particularly suitable for learning sequential patterns over multiple scales, and semi-markov HMM and CRFs [9] in which the base methods are modified to allow better modelling of state durations. The related but representationally more powerful Conditional Random Fields (CRFs) are commonly used in sequence labelling tasks, and have been used for activity recognition in [10] and [11]. This method is covered in detail in Section III.

A Support Vector Machine (SVM) uses the Kernel trick to transform the data to be classified into a high-dimensional space, in which a maximum-margin separating hyperplane is learnt. In [6], an SVM is used on features describing a window of sensor data, which includes the time of day, duration of the event sequence comprising the window, and the number of activations for each sensor. A variant of the SVM called the structural SVM presented in [12] extends the SVM by modifying the optimisation problem the SVM solves so that a sequence pair $(\mathbf{x}, \mathbf{y})$ as a whole is considered as a single training example, instead of treating each (x, y) pair in each sequence as a separate training example.

III. Conditional Random Fields

Conditional Random Fields (CRFs) are a discriminative method for structured prediction [13]. This means that instead of modelling the joint probability $P(Y, X)$ (allowing us in theory to generate samples of X,Y pairs) we model the conditional probability $P(Y|X)$, which allows us only to discriminate between different labellings Y given an observation sequence X. A major advantage of this approach is that dependencies between observables need not be modelled, an aspect of learning generative models which can be problematic.

Structured prediction methods model a relationship between variables whose dependencies can be represented by a graph. In this paper, we will be concerned with the special case of linear chain CRF, where the graph structure can be represented as in Fig. 1. Each observed sequence is an X, Y pair of observation X and label Y sequences, with x_t and y_t denoting the observation and label at position t in the sequence, respectively.

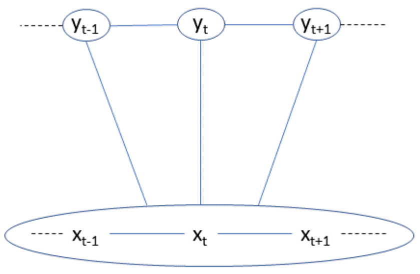


Fig. 1: Structure of Linear Chain CRF

In a linear chain CRF the conditional probability is modelled as:

$$P(Y|X) = \frac{1}{Z} exp\left[\sum_t \sum_i w_i f_i(y_t, y_{t-1}, X)\right]$$

where Z is a normalisation defined as:

$$Z = \sum_{\mathbf{y}} exp\left[\sum_t \sum_i w_i f_i(y_t, y_{t-1}, X)\right]$$

with the first sum here over all possible labellings. The features f_i are functions which represent knowledge about how desired labellings for the model look, and can depend on the label at position t, the label at the previous position $t-1$, and the whole observation sequence X. For example, a simple feature $f = I(y_t = y_{t-1})$ where I is the indicator function would, assuming positive associated weight w_i, represent a continuity condition. That is, if the last observation in the sequence at $t-1$ had label 'making lunch', there is a good reason to believe we could still be making lunch at time t. The features are typically binary, though this is not a requirement and some application areas do use non-binary features. The weight w_i encodes the importance of satisfying this feature (or avoid it given a negative weight). In the case of linear chain CRFs, which we are concerned with, inference can be done using generalisations of the forward-backward and Viterbi algorithms used for HMMs [13]. A CRF is usually trained by maximising the likelihood:

$$l(\mathbf{w}) = p(Y^{(i)}|X^{(i)}, \mathbf{w}) \tag{1}$$

$$= \sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{k=1}^{K} w_k f_k(y_t^{(i)}, y_{t-1}^{(i)}, X_t^{(i)}) - \sum_{i=1}^{N} logZ(X^{(i)}). \tag{2}$$

Maximising this value provides us with the parameter setting for which the observed data has the highest probability. The optimisation of $l(\mathbf{w})$ can be done using a variety of standard techniques for numerically optimising non-linear functions. Quasi-Newton methods such as BFGS [14] are commonly used in the case of CRFs as they are well suited in the case where the number of parameters may be large. In practice, the likelihood is often augmented with a regularization term which penalises overly large parameter values.

A strength of CRF is its ability to incorporate a wide variety of arbitrary features into the model, and learn their relative importance. But, this leaves open the question of which features are best to use. Typically, a number of atomic tests are defined (such as $I(y_t = \hat{y})$, $I(x_{t-1} = \hat{x})$, etc) and features are made following a number of templates [13]. All features which adhere to this template are generated. These templates take the form of simple conjunctions of these atomic tests, for example

$$I(y_t = y)I(y_{t-1} = y\prime) \qquad \forall y, y\prime \in \mathcal{Y}$$

which would generate $|\mathcal{Y}|^2$ features each of which is non-zero only for one specific combination of labels at steps t and $t-1$.

A major problem of this approach is that if we wish to include more complex conjugations, we must include more complex templates meaning the number of features can become extremely large (for example [15] uses millions of features), making learning computationally expensive or even intractable. There is also the danger of overfitting in a model with such a large number of paramters. There have been some attempts at a more principled generation of features, the most notable of which is the method presented in [16]. There, an iterative approach is taken, where at each iteration a set of candidate features are generated, and a number of these are added to the model based on an estimate of the effect of their inclusion on the log-likelihood of the correct sequence. Other methods rely on heuristics or domain knowledge to guess at some useful complex features. To overcome these problems, the presented approach augments simple template features with more complex ones extracted from an ensemble (random forest) of decision trees.

IV. Feature induction via Decision Trees

Typically, as explained above, a CRF uses a large set of features consisting of relatively simple conjunctions of atomic tests. If we consider a decision tree built to predict the current label y_t given the previous label y_{t-1} and the most recent N observations X_t, then a leaf of the decision tree defines a conjunction of exactly the form we wish to generate for the CRF. The path to the leaf is a conjunction of atomic tests on the $x_k \in X_t$ and y_{t-1}. Together with the indicator $I(y_t = y_l)$, where y_l is the label assigned to the leaf node, this defines a conjunction of the desired form as shown in Fig. 2. These conjunctions can then be extracted and used as features for a CRF. The great advantage of this approach is that significantly more complex features can be used, as we are not generating them blindly, as we would via templates, but rather are generating them based on the data, seeking useful conjunctions in identifying relationships within the data. We can also control the complexity of the features generated by adjusting the pruning level applied to the tree.

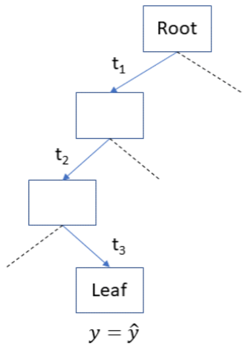


Fig. 2: Illustration of tree path, generating a feature which is a conjunction of the tests at each split and the indicator of the leaf label, $f = t_1 t_2 t_3 I(y = \hat{y})$.

A. Random Forests

Decision trees are an *unstable* classifier [17], which means that a very small change in the training data or learning parameters of the model can result in a very different learned model. In extracting features from only one tree, we may miss many other equally good conjunctions that could have appeared had the data only been very slightly different. In the Classifier Fusion literature, there are methods which exploit this behaviour to produce ensemble methods that greatly outperform any of its component models. One of these is Random Forests [18]. In this method, a number of parameters of the decision tree that control learning are placed in a parameter vector θ, and an ensemble of trees is learned using a randomly generated parameter vector θ_k for each tree. This learning parameter vector usually defines a sampling of the data to use during training (often a bootstrap sampling), and a subset of observations to be considered for each split of the tree. These trees are combined via a majority vote. We will use the same approach to create a diverse ensemble of trees from which to extract features, to ensure a wide variety of useful features are generated. When building a Random Forest for classification, the component trees are usually fully grown [18] (i.e no pruning applied) however we will depart from this including the pruning level as one of the parameters in θ_k. The motivation for this is that we expect including a wide variety of feature complexities (conjunction lengths) in our feature set to be beneficial.

B. Extracting CRF features from a Random Forest

Assume we have a basic set of features $\mathcal{B}$. We also have as a base learner a decision tree whose learning is controlled by a parameter vector θ. In our case, this vector $\theta = [S, \beta, \alpha]$ consists of a bootstrap sampling S over the training data, a sequence β of randomly generated subsets of size M of the features in $\mathcal{B}$ that will be considered for each split of the tree, and a parameter α controlling the level of pruning. An ensemble of N trees is trained, each using a different randomly generated θ.

Each split of a tree is a binary test t on one of these features. For an example x at a node, if this test is true, the example passes to the left child node, if its negation is true it passes instead to the right child node. Starting at the root node, the sequence of tests leading to a leaf node defines a path $p = \{t_k\}$, $k = 1, d$ of depth d, with leaf label $y^{(l)}$. From this we define a feature $f = I(y = y^{(l)}) \prod_{k=1}^{d} t_k$, which in the CRF context is a binary feature returning one for all labellings where an example would reach the leaf and has the same label as the leaf. We traverse the tree and extract a feature as above for each leaf of the tree, repeating for each tree in the ensemble. This results in a set of conjugations $\mathcal{F}$ which will be used as the features of the CRF. We also add the following simple template-based features to $\mathcal{F}$:

$$I(x_t = \hat{x}) I(y_t = \hat{y}) \qquad \forall\, \hat{x} \in \mathcal{X}, \hat{y} \in \mathcal{Y} \tag{3}$$

$$I(x_{t-1} = \hat{x}) I(y_t = \hat{y}) \qquad \forall\, \hat{x} \in \mathcal{X}, \hat{y} \in \mathcal{Y} \tag{4}$$

$$I(y_t = y_{t-1}) \tag{5}$$

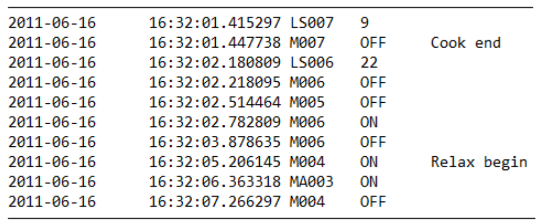
This feature set is then used to train a CRF. The method is summarised in pseudocode in Program 1.

Program 1 Pseudocode for proposed method

```
initialise empty feature set $\mathcal{F}$
for each tree $k$ of $N$
    initialise parameter vector $\theta_k$ randomly
    train tree $h_k$ using $\theta_k$
    for each leaf $l$ in $h_k$
        extract path to leaf $p_l$
        create feature as conj. $f_{lk} = p_l * I(y_t = y_l)$
        add feature $f_{lk}$ to $\mathcal{F}$
    end
end
if desired, add additional template features to $\mathcal{F}$
train CRF on $\mathcal{F}$
```

V. Experiments

The dataset used in this paper stems from the CASAS project [6], and is freely available online [7]. The dataset is collected over a period of a few months from a smart apartment housing a single resident. The apartment is outfitted with a variety of sensors including motion, temperature, light, and power usage sensors, door sensors, and a small number of tagged objects. The data takes the form of a sequence of sensor activations, with some time intervals annotated with the activity that was being performed, for example, `Making_Lunch`, `Take_Medication`, or `Sleep`. A small excerpt of the raw data can be seen in Fig. 3. There are 29 distinct activities recorded in total, with some appearing significantly more often than others. The dataset consists of 127 possible sensor activations. Each activation consists of an ID, a timestamp, and a reading, which takes either a boolean ON/OFF (for door, motion sensors etc.) or real (for temperature, etc.) value. Some of the sensor activations are various battery status indicators for the sensors; we discard these as they are not useful for the



```
2011-06-16      16:32:01.415297 LS007   9
2011-06-16      16:32:01.447738 M007    OFF     Cook end
2011-06-16      16:32:02.180809 LS006   22
2011-06-16      16:32:02.218095 M006    OFF
2011-06-16      16:32:02.514464 M005    OFF
2011-06-16      16:32:02.782809 M006    ON
2011-06-16      16:32:03.878635 M006    OFF
2011-06-16      16:32:05.206145 M004    ON      Relax begin
2011-06-16      16:32:06.363318 MA003   ON
2011-06-16      16:32:07.266297 M004    OFF
```

Fig. 3: Example of raw data in CASAS dataset.

classification task. This leaves us with 67 sensor activations overall. For each sensor, we build two features at a time t:

1) Given a time window T_w, we calculate the average of the sensor activation over T_w, $x_t = \frac{1}{T_w}\int_{t-T_w}^{t} x dt$
2) The time since the sensor was last on (for binary sensors) or last changed (for real-valued sensors)

We also add a final feature, which is a time of day feature taking a real value between 0 and 1. This gives a set of 135 basic features on which we build our models. We parse the raw data file to build these features, and sample them every minute, together with the corresponding label, to give the dataset used in our experiments. For comparison purposes, below, we present the results for both the method proposed in Section IV and other commonly used methods.
Decision Tree: A single decision tree grown using the information gain split criterion and pruned using cost-complexity pruning.
CRF-Template: A CRF model built using a set of simple template based features. The features used are as described in Section IV-B.
CRF-Forest: A CRF built using features extracted from a Random Forest as described in Section IV.
CRF-Both: A CRF built using features extracted from a Random Forest augmented by simple template based features, i.e. using the features from both of the above two methods.
SVM: A Multi-class SVM consisting of an ensemble of 1 vs 1 SVM classifiers. There are $\frac{1}{2}K(K-1)$ binary SVMs in total. Each member of the ensemble is trained using a class i as positive examples, and a class j as negative examples, with the remainder ignored. A member is trained for each possible (unordered) i, j pair.
Random Forest: An ensemble of 200 fully grown decision trees, each trained on a bootstrap sampling of the training set and splitting on a randomly selected 40% of the available features at each split. The optimal split is determined using the information gain criterion.

Method	Accuracy
Decision Tree	65.2
CRF-Template	55.38
CRF-Forest	72.84
CRF-Both	62.7
SVM	65.42
Random Forest	70.9

TABLE I: Average accuracy (%) on activity recognition task

The results of these methods on the CASAS dataset are shown in Table I, denoting that the use of diverse but highly relevant complex features extracted from a random forest produces a much better performing CRF model than one built on simple template features. The use of a CRF on features generated from the random forest trees provided a performance increase compared to using the trees as an ensemble classifier, as the sequential nature of the problem is better taken into account. Also, the ability of the CRF to learn weightings for features provides an additional layer of subtlety that the ensemble method cannot. Something which was a little surprising was the lesser performance of the CRF built on the combination of tree features and simple template features, compared to using tree features only. This could happen if the simple template features added contained little information that was not already contained in the more complex features from the random forest. In that case, the additional features would unnecessarily complicate the optimisation process and may have also lead to some overfitting.

VI. Conclusion

In this paper, we have introduced an approach to feature generation for CRFs which exploits the instability of decision trees to produce a diverse set of complex features which are highly targeted to be relevant to the classification problem to be solved, through the same approach as that used in building random forest ensemble classifiers. In contrast to complex template features which produce large numbers of mostly irrelevant features, our method produces a smaller number of features which are well suited to the classification task at hand. Further, the performance was improved compared to both the CRF built on simple template-based features and to the decision trees used as a random forest ensemble, showing that these two aspects of our method complement each other well. Future work could investigate in depth how controlling the complexity of the features extracted from the random forest using pruning of the resulting decision trees, would affect the CRF models build on them.

References

[1] A. Konios et al., "Unifying and Analysing Activities of Daily Living in Extra Care Homes", in Proceedings of 3rd Computing and Cyber Science and Technology Congress, 2018.

[2] Y. Jing et al., "An intelligent well-being monitoring system for residents in extra care homes", in Proceedings of the 1st Intl Conf on Internet of Things and Machine Learning, 2017.

[3] J. D. Lafferty, et al., "Conditional random fields: Probabilistic models for segmenting and labeling sequence data", in Proceedings of the 18th Intl Conf on Machine Learning, pp. 282 - 289, 2001.

[4] J. Esch, "A survey on ambient intelligence in healthcare", Proceedings of the IEEE, vol. 101, no. 12, pp. 2467 - 2469, 2013.

[5] P. Woznowski et al., "A multi-modal sensor infrastructure for healthcare in a residential environment", in 2015 IEEE International Conference on Communication Workshop (ICCW), pp. 271 - 277, 2015.

[6] D. J. Cook et al., "Casas: A smart home in a box", Computer, vol. 46, no. 7, pp. 62 - 69, 2013.

[7] Casas datasets, no. 24, http://casas.wsu.edu/datasets/, 2015.

[8] S. Fine et al., "The hierarchical hidden markov model: Analysis and applications", Mach. Learn., vol. 32, no. 1, pp. 41 - 62, 1998.

[9] T. van Kasteren, G. Englebienne, and B. J. A. Kröse, "Activity recognition using semi-markov models on real world smart home datasets", JAISE, vol. 2, no. 3, pp. 311 - 325, 2010.

[10] M. Agarwal and P. Flach, "Activity recognition using conditional random field", in Proceedings of the 2nd Intl Workshop on Sensor-based Activity Recognition and Interaction, pp. 4:1 - 4:8, 2015.

[11] E. Nazerfard et al., "Conditional random fields for activity recognition in smart environments", in Proceedings of the 1st ACM International Health Informatics Symposium, ser. IHI 10. pp. 282 - 286, 2010.

[12] I. Tsochantaridis et al., "Large margin methods for structured and interdependent output variables", J. Mach. Learn. Res., vol. 6, pp. 1453 - 1484, 2005.

[13] C. Sutton and A. McCallum, "An introduction to conditional random fields", Found. Trends Mach. Learn., vol. 4, no. 4, pp. 267 - 373, 2012.

[14] J. Nocedal and S. J. Wright, Numerical Optimization, 2nd ed., Springer, 2006.

[15] F. Sha and F. Pereira, "Shallow parsing with conditional random fields", in Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1, ser. NAACL 03. pp. 134 - 141, 2003.

[16] A. McCallum, "Efficiently inducing features of conditional random fields", in Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence, ser. UAI03. pp. 403 - 410, 2003.

[17] R.-H. Li and G. G. Belford, "Instability of decision tree classification algorithms", in Proceedings of the 18th ACM SIGKDD Intl Conf on Knowledge Discovery and Data Mining, pp. 570 - 575, 2002.

[18] L. Breiman, "Random forests", Mach. Learn., vol. 45, no. 1, pp. 5 - 32, 2001.