Activity Context Representation: Techniques and Languages
AAAI Technical Report WS-12-05

Challenges in Learning Optimum Models for Complex First Order Activity Recognition Settings

Naveen Nair
IITB-Monash Research Academy
Dept. of CSE, IIT Bombay
Faculty of IT, Monash University
naveennair@cse.iitb.ac.in

Amrita Saha
Dept. of CSE,
IIT Bombay
amrita@cse.iitb.ac.in

Ganesh Ramakrishnan
Dept. of CSE, IIT Bombay
IITB-Monash Research Academy
ganesh@cse.iitb.ac.in

Shonali Krishnaswamy
Institute for Infocomm Research (I2R), Singapore
Faculty of IT, Monash University
IITB-Monash Research Academy
Shonali.Krishnaswamy@monash.edu

Abstract

Non intrusive activity recognition systems typically read values from sensors deployed in an environment and combine them with user annotated activities to build a probabilistic model. Recently, features constructed from activity specific conjunctions of binary sensor values have been shown to improve the classification accuracy. Such systems employ greedy feature induction techniques to find the observation features and combine them with state transition distribution in a Hidden Markov Model or a Conditional Random Field. An exhaustive search for optimum features is infeasible in this exponential feature space. We have recently extended the rule ensemble learning using hierarchical kernels (RELHKL) framework, that learns a sparse set of simple features and their optimum weights, to structured output spaces for learning optimum observation features along with the transition features and their weights. The exponentially large space of conjunctions is handled efficiently by exploiting its hierarchical structure. Our experiments have shown good improvement over other approaches. Although such approaches solve propositional classification problems optimally, their first-order extension is non-trivial and is a challenging problem. In this paper, we discuss about the challenges involved in leveraging the RELHKL in structured output spaces approach to learn optimum features in complex first order activity recognition settings.
Keywords: *activity recognition, first order logic, hidden markov models, hierarchical kernels, rule learning, structured output spaces, support vector machines.*

1 Introduction

Activity recognition systems in home environments typically recognize activities performed by inhabitants and help other components of smart homes to perform functionalities such as optimization of house hold equipment usage or monitoring of health condition of elderly people living alone (as activities of daily living are indicators of health condition of old age people (Wilson 2005; van Kasteren et al. 2008; Gibson, van Kasteren, and Krose 2008)). Such non-intrusive settings typically have sensors installed at various locations in a home. A probabilistic model is learned during the training period from the sensor values and the activities annotated by the user. During the inference phase, the user activities are estimated by the probabilistic model, based on the observed sensor values. Many of the existing systems pose and solve this and similar problems in propositional settings (van Kasteren et al. 2008; Gibson, van Kasteren, and Krose 2008; Tsochantaridis et al. 2004; Tsochantaridis 2006; McCallum 2003; Nair, Ramakrishnan, and Krishnaswamy 2011; Nair et al. 2012). In settings that have complex first order relations (first order predicates and relations represent common properties of groups of objects and the interactions between these predicates respectively), systems that solve the problem in propositional space may not lead to a good model. We first briefly discuss propositional activity recognition settings followed by discussion on first order settings and finally the challenges involved in learning optimum first order models.

In typical deployments, sensor values are recorded at regular time intervals. The joint state of these sensor values at each time t form our observations and we will represent them as $\mathbf{x}_t$. The user activity at time t forms the hidden state, which we represent by y_t. The history of sensor readings and the corresponding activities (as manually identified later) can be used to train prediction models such as the Hidden Markov Model (HMM) (Rabiner 1989), the Conditional Random Field (CRF) (Lafferty, McCallum, and Pereira 2001) or SVM on Structured Output Spaces (StructSVM) (Tsochantaridis 2006; Tsochantaridis et al. 2004), which could be later used to predict activities based on sensor observations. These approaches typically assume that y_t at time t is independent of all previous activities given y_{t-1} at time $t-1$ and $\mathbf{x}_t$ at time t is independent of all other variables given y_t. Prediction involves determining the label (activity) sequence that best explains the observation (joint state of sensors) sequence using dynamic programming (Forney 1973). In the rest of this paper, we assume sensor values as binary values. In other types of deployments, such as discrete and continuous value sensors, one or more binary values can be constructed from a single discrete/continuous valued output.

Activity recognition datasets tend to be sparse; that is, one could expect very few sensors to be on at any given time instance. Moreover, in a setting such as activity recognition, one can expect certain combinations of (sensor) readings to be directly indicative of certain activities. While HMMs, CRFs and StructSVM attempt to capture these relations indirectly, we have illustrated that discovering activity specific conjunctions of sensor readings (as features) can improve the accuracy of prediction (Nair, Ramakrishnan, and Krishnaswamy 2011). Earlier, McCallum (2003) followed a similar approach for inducing features for a CRF model. However, both these approaches greedily search the space of conjunctions, since an exhaustive search for the optimal features is exponential in the number of basic features (sensors in the case of activity recognition). Rule Ensemble Learning using Hierarchical Kernels (RELHKL) is one existing approach that can handle exponential feature space. However, RELHKL is specific to single variable binary classification problems and is not applicable to multi class structured output space classification problems. We have recently proposed a generalization of RELHKL to multi class structured output spaces such as sequence prediction problems (Nair et al. 2012). We refer to this approach as StructRELHKL. StructRELHKL builds on the StructSVM (Tsochantaridis et al. 2004; Tsochantaridis 2006) framework, where emission (observation-activity relationship) and transition (activity-activity relationships in successive time steps) relations are modeled as features of SVM, and use a hierarchical regularizer on emission features to select a small set of simple conjunctions. The loss function used is same as that of StructSVM. We have demonstrated through experiments that StructRELHKL selects a small set of simple features and outperform other existing approaches in terms of prediction accuracy. Please refer (Nair et al. 2012) for a comparison of StructRELHKL and other approaches.

The above approaches have limitations that they are specific to propositional feature settings and are unable to handle first order structures. We now give a brief introduction to first order settings and the limitations of the above approaches in such settings.

In a first order setting, predicates are used to attribute properties to objects and can be used to represent groups of objects with same properties. For example, objects such as *bread, butter, sugar, etc.* are items used for break fast and *rice, vegetables, etc.* are items used for lunch. A predicate *breakFastItem(X)* says that *X* is a break fast item. A ground instance of this can be *breakFastItem(butter)* and is a true ground atom. Whereas, *breakFastItem(rice)* gives false as rice is not a break fast item. Generalized features constructed using such first order predicates help to represent and interpret the environment in an efficient way. For example, *activity(T,prepareBreakFast):-microwave(T,X),breakFastItem(X)* says that the activity at time *T* is *prepareBreakFast* if *microwave* has object *X* in it at time *T* and *X* is a *breakFastItem*. There are many first order systems that learn such features. However, most of these systems learn the features by searching in a lattice (partial ordered set with a unique top and a unique bottom) of rules/clauses greedily by pruning away branches based on the coverage of examples. However, since an exhaustive search in the super exponential space is infeasible, an optimum model is not guaranteed.

In activity recognition problems, the transition features also have to be added to the set of first order emission features (features and rules are synonymously used in this paper, as the condition part of any rule can be treated as an activity specific feature). As StructRELHKL is shown to be learning optimum set of features in an exponential feature space for structured output classification problems, it is intuitive to think of grounding all the predicates with all possible instances of variables and learn StructRELHKL model on this. However, the number of groundings itself can be huge and thus can be infeasible in real wold settings. Moreover the model learned will not be a generalized one. In this paper we look into the challenges involved in learning an optimum first order model by extending StructRELHKL.

The paper is organized as follows. Propositional greedy feature induction approaches and hierarchical kernels based approaches are discussed in Section 2. Section 3 discusses about the first order rule learning settings. We discuss about the challenges in learning optimum features for first order settings in Section 4 and conclude the paper in Section 5.

2 Feature Induction in Propositional Settings

In this section, we first give a brief description of greedy feature induction approaches for structured output classification problems. We then introduce the recent works on learning optimal features using hierarchical kernels on structured output spaces.

Feature Induction for Structured Outputs

In this subsection, we discuss our approach on feature induction for HMM (Nair, Ramakrishnan, and Krishnaswamy 2011) as well as feature induction for CRF by McCallum *et al.* (2003). Both the approaches propose feature induction methods that iteratively construct feature conjunctions that increase an objective. These approaches start with no features and at each step, consider a set of candidate features (conjunctions or atomic). The features, whose inclusion will lead to maximum increase in the objective are selected. Weights for the new features are trained. The steps are iterated until convergence. While McCallum *et al.* (2003) trains a CRF model and uses conditional log-likelihood as the objective for the greedy induction, we train an HMM and use prediction accuracy on a held out dataset (part of the training data) as the objective. This effectively solves the problem of incorrect conditional independence assumption among the sensor values given an activity while not dealing with exponential observation space.

Although these greedy feature induction approaches have been shown to improve performance, they cannot guarantee an optimal solution. An exhaustive search to find the optimal solution is expensive due to the exponential size of the search space. We next discuss an approach for optimal induction of feature conjunctions.

Optimal Feature Induction using Hierarchical Kernels

In this subsection, we briefly discuss our work on rule ensemble learning using hierarchical kernels in structured output spaces (Nair et al. 2012).

Our work builds on StructSVM framework (Tsochantaridis et al. 2004; Tsochantaridis 2006) which learns a classification model for structured outputs such as sequence, trees, graphs etc. We discuss our approach in the context of sequence prediction problems in the activity recognition domain.

The objective of learning with structured output spaces is to learn functions of the form $\mathcal{F} : \mathcal{X} \rightarrow \mathcal{Y}$ from training data, where $\mathcal{X}$ and $\mathcal{Y}$ are input and output sequence spaces respectively. A discriminant function $F : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ is learned from training data that consists of pairs of input and output sequences. The prediction is performed using the decision function $\mathcal{F}(X; \mathbf{f})$:

$$\mathcal{F}(X; \mathbf{f}) = \arg\max_{Y \in \mathcal{Y}} F(X, Y; \mathbf{f}), \tag{1}$$

where $F(X, Y; \mathbf{f}) = \langle \mathbf{f}, \boldsymbol{\psi}(X, Y) \rangle$ represents a score which is a scalar value based on the features $\boldsymbol{\psi}$ involving input sequence X and output sequence Y values and parameterized by a parameter vector $\mathbf{f}$. In the case of sequence prediction, features are constructed to represent emission (observation) and transition distributions. Unlike StructSVM, our emission features represent all possible conjunctions of sensor values for each activity. The objective for training is to find a sparse set of simple features $\boldsymbol{\psi}(X, Y)$ and their optimal weights $\mathbf{f}$, which make the score of original sequences greater than any other possible sequence by a good margin.

Loss functions in structured outputs have to measure the amount by which the prediction deviates from the actual value and hence the zero-one classification loss is not sufficient. We use the micro-average of wrong predictions as loss function in our derivations. A loss function is represented as $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$. $\Delta(Y, \hat{Y})$ is the loss value when the true output is Y and the prediction is $\hat{Y}$. Before going into further details of the model, we introduce the following notations.

Let the input/observation at p^{th} time step of the i^{th} sequence (example) be $\mathbf{x}_i^p$, where $\mathbf{x}_i^p$ is a vector of binary sensor values (each element of the vector represents the value of a sensor fixed at a location such as groceries cupboard, bathroom door etc. at that time step). Similarly, output (activity) at p^{th} time step of the i^{th} example is represented by y_i^p. Let y_i^p can take any of n values. A feature vector, $\boldsymbol{\psi}$, contains entries for emission/observation and the transition distribution. To learn the emission structure, the feature vector has to be modified to include the emission lattice defined in (Nair, Ramakrishnan, and Krishnaswamy 2011). The emission lattice has conjunctions of basic features (sensors) as nodes and obeys a partial order. The top node is the empty conjunction and the bottom node is the conjunction of all the basic features. The nodes at level 1, denoted by B, are basic features themselves. As followed in (Jawanpuria, Jagarlapudi, and Ramakrishnan 2011), $D(v)$ and $A(v)$ represent the set of descendants and ancestors of the node v in the lattice. Both $D(v)$ and $A(v)$ include node v. The hull and the sources of any subset of nodes $\mathcal{W} \subset \mathcal{V}$ are defined as $hull(\mathcal{W}) = \bigcup_{w \in \mathcal{W}} A(w)$ and $sources(\mathcal{W}) = \{w \in \mathcal{W} | A(w) \bigcap \mathcal{W} = \{w\}\}$ respectively. The size of set $\mathcal{W}$ is denoted by $|\mathcal{W}|$. $\mathbf{f}_\mathcal{W}$ is the vector with elements as $f_v, v \in \mathcal{W}$. Also let the complement of $\mathcal{W}$ denoted by $\mathcal{W}^c$ be the set of all nodes belonging to the same activity that are not in $\mathcal{W}$. For the sake of visualization, we assume there is a lattice for each label. Therefore, elements of $\boldsymbol{\psi}$ vector correspond to the nodes in the conjunction lattice of each label and the transition features. We represent the emission and transition parts of the vector $\boldsymbol{\psi}$ as $\boldsymbol{\psi_E}$ and $\boldsymbol{\psi_T}$ respectively. We assume that both $\boldsymbol{\psi_E}$ and $\boldsymbol{\psi_T}$ are of dimension equal to the dimension of $\boldsymbol{\psi}$ with zero values for all elements not in their context. That is, $\boldsymbol{\psi_E}$ has dimension of $\boldsymbol{\psi}$, but has zero values corresponding to transition elements. In similar spirit, we split the feature weight vector $\mathbf{f}$ to $\mathbf{f_E}$ and $\mathbf{f_T}$. Similarly, $\mathcal{V}$, the indices of the elements of $\boldsymbol{\psi}$, is split into $\mathcal{V}_\mathbf{E}$ and $\mathcal{V}_\mathbf{T}$.

To learn a sparse emission model, we employ a ρ-norm regularizer (Jawanpuria, Jagarlapudi, and Ramakrishnan 2011) on the emission features. Since, the transition features are not exponential and do not obey a partial order, we impose a 2-norm regularizer. Therefore, we separate the regularizer term in the SVM objective into those corresponding to emission and transition features and construct the following SVM formulation,

$$\begin{aligned} &\min_{\mathbf{f}, \boldsymbol{\xi}} \frac{1}{2}\Omega_E(\mathbf{f_E})^2 + \frac{1}{2}\Omega_T(\mathbf{f_T})^2 + \frac{C}{m}\sum_{i=1}^{m} \xi_i, \\ &\forall i, \forall Y \in \mathcal{Y} \setminus Y_i : \langle \mathbf{f}, \boldsymbol{\psi}_i^\delta(Y) \rangle \geq 1 - \frac{\xi_i}{\Delta(Y_i, Y)} \\ &\forall i : \xi_i \geq 0 \end{aligned} \tag{2}$$

where $\Omega_E(\mathbf{f_E})$ is defined in (Jawanpuria, Jagarlapudi, and Ramakrishnan 2011) as $\sum_{v \in \mathcal{V}_\mathbf{E}} d_v \parallel \mathbf{f}_{\mathbf{E}D(v)} \parallel_\rho, \quad \rho \in [1, 2], \quad d_v \geq 0$ is a prior parameter showing usefulness of the feature conjunctions, $\mathbf{f}_{\mathbf{E}D(v)}$ is the vector with elements as $\parallel f_{Ew} \parallel_2 \ \forall w \in D(v)$, and $\parallel . \parallel_\rho$ represents the ρ-norm , $\Omega_T(\mathbf{f_T})$ is the 2-norm regularizer $\left(\sum_i f_{Ti}^2\right)^{\frac{1}{2}}$, m is the number of examples, C is the regularization parameter, ξ's are the slack variables introduced to allow errors in the training set in a soft margin SVM formulation, and $X_i \in \mathcal{X}$ and $Y_i \in \mathcal{Y}$ represent the i^{th} input and output sequence respectively. $\langle \mathbf{f}, \boldsymbol{\psi}_i^\delta(Y) \rangle$ represents the value $\langle \mathbf{f}, \boldsymbol{\psi}(X_i, Y_i) \rangle - \langle \mathbf{f}, \boldsymbol{\psi}(X_i, Y) \rangle$.

The 1-norm in $\Omega_E(\mathbf{f_E})$ forces many of the $\parallel \mathbf{f_E}_{D(v)} \parallel_\rho$ to be zero. Even in cases where $\parallel \mathbf{f_E}_{D(v)} \parallel_\rho$ is not forced to zero, the ρ-norm forces many of node v's descendants to zero. This ensures a sparse and simple set of features. The above SVM setup has two inherent issues which makes it a hard problem to solve. The first is that the regularizer, $\Omega_E(\mathbf{f_E})$, consists of ρ-norm over descendants of each lattice node, which makes it exponentially expensive. The second problem is the exponential number of constraints for the objective. Next paragraph discusses how to solve the problem efficiently.

By solving (2), we expect most of the emission feature weights to be zero. As illustrated by Jawanpuria *et al.*, the solution to the problem when solved with the original set of features is the same but requires less computation when solved only with features having non zero weights at optimality. Therefore, an active set algorithm can be employed to incrementally find the optimal set of non zero weights (Jawanpuria, Jagarlapudi, and Ramakrishnan 2011). In each iteration of the active set algorithm, since the constraint set in (2) is exponential, a cutting plane algorithm has to be used to find a subset of constraints of polynomial size so that the corresponding solution satisfies all the constraints with an error not more than ϵ. In our work (Nair et al. 2012), we derive a dual for (2) with the feature set reduced to active features and a sufficient condition to check for optimality. The active set algorithm invokes a mirror descent algorithm and the mirror descent algorithm invokes a cutting plane algorithm. Please refer to (Nair et al. 2012) for more details.

3 First Order Rule Learning Settings

In this section, we first discuss the limitations of propositional logic and then introduce first order logic in the context of activity recognition. We then briefly discuss some of the properties and methodologies in first order learning settings.

In an activity recognition system, readings from sensors installed at different locations are recorded. These combined with the user annotated activities are used to build a probabilistic model. Approaches such as HMM, CRF, feature induction assisted HMM, CRF with feature induction, StructRELHKL etc. model each sensor as a propositional feature. These approaches have the limitation that they are not capable to represent groups of objects with same properties efficiently or capture the relationships among those groups. For example, consider a kitchen environment where sensors or RFID tags are attached to many objects such as *rice container, wheat container, sugar container, salt container, cheese, bread, butter, microwave, gas stove, electric stove, mixer, fridge etc*. Let the activities being performed are *prepare break fast, prepare lunch and prepare dinner*. Intuitively, a person preparing break fast may place some break fast food items on the *stove* and therefore, knowing the break fast food items (*bread, butter, milk etc.*) would make it easy to conclude that the person is preparing break fast. However, propositional models lack this expressive power and result in exponential increase in features leading to an infeasible model in many settings. We now discuss how first order logic helps to overcome this limitation.

For compact representation, better interpretation, and efficient inference, objects are attributed with properties and groups of objects with same properties can be represented in a general form in first order logic. The property attributed to objects and used to group similar objects is called a first order predicate and it holds a meaning. For example, objects such as *bread, butter, sugar, etc.* are items used for break fast and *rice, vegetables, etc.* are items used for lunch. Predicates *breakFastItem(X)* and *lunchItem(Y)* stands for "*X* is a break fast item" and "*Y* is a lunch item" respectively. Here *X* and *Y* are typed variables. A ground instance of this can be *breakFastItem(butter)* and is a true ground atom. Whereas, *breakFastItem(rice)* is a false ground atom, as rice is not a break fast item according to the definition. Predicates can also represent relations between two or more objects. For example, *near(X,Y)* stands for "object *X* is near to object *Y*". First order logic also enables to construct features based on properties and provides a meaningful definition of the world. Generalized features constructed using first order predicates help to represent and interpret the environment in an efficient way. For example, *activity(T,prepareBreakFast):-microwave(T,X),breakFastItem(X)* says that the activity at time *T* is *prepareBreakFast* if *microwave* has object *X* in it at time *T* and *X* is a *breakFastItem*. In real world settings, possible groundings can be too large and thus propositional approaches have their limitations. Inductive logic programming (ILP) (Nienhuys-Cheng and Wolf 1997) approaches learn first order relations from background knowledge, positive examples and negative examples. Background knowledge can be in the form of true predicates or rules known a priori. Although ILP systems which attribute probabilistic information to the clauses (clauses are rules represented in the form of disjunctions of literals) are called probabilistic ILP, we use the term ILP in a general sense. Aleph (Srinivasan 2007), alchemy (Domingos et al. 2006), FOIL (Quinlan and Cameron-Jones 1993) etc. are some of the ILP systems. Works by Landwehr et al. (2009), Kersting et al. (2006) and Natarajan et al. (2008) are a few relational models introduced in activity recognition domain. ILP systems learn the features by searching in a lattice (partial ordered set with a unique top and a unique bottom) of clauses (features). These systems prune away branches based on some heuristics and explores the exponential lattice in an efficient way. The heuristic, in most cases, is a score which is a function of positive and negative examples covered by the current model. For instance, Aleph is an ILP system that in each iteration, selects a positive example, builds the most-specific clause based on the example, searches for a more general clause that has the best score, and removes examples made redundant by the current clause. However, the search for the best set of features is based on heuristics and thus an optimum model is not guaranteed. An exhaustive search in the super exponential space is infeasible. Inference in these systems involves finding an interpretation (truth assignments to all ground predicates) that maximizes the number (possibly weighted sum) of satisfied clauses. We now discuss how ILP systems exploit the structure of search space to learn general clauses efficiently.

As mentioned in the previous paragraphs, the objective of ILP systems is to automatically construct general clauses, that gives a better understanding about the world, from specific ones. A clause C_1 is regarded to be more general than C_2 if and only if $C_1 \models C_2$. Therefore, the relation $\models$ can be used to make an ordering over a set of clauses. ILP systems search such orderings of clauses to find general clauses. Since lattices are well structured orderings, many ILP systems use lattices to represent all the possible clauses. We discuss subsumption lattice and implication ordering, two generality orderings profoundly used in ILP, in the following paragraphs.

A clause C subsumes a clause D if there is a substitution θ such that $C\theta \subseteq D$ and therefore $C \models D$. $C\theta \subseteq D$ indicates that, after applying substitution θ to C, every literal in C appears in D (Khardon and Arias 2006; Nienhuys-Cheng and Wolf 1997). For example, let C : *cooking(T,X):-microwave(T,X)* subsumes D : *cooking(T,rice):-microwave(T,rice),lunchItem(rice)*, where $\theta = \{X/rice\}$. This helps to order clauses in a quasi-order (reflexive and transitive) which can be viewed as a partial order among equivalence partitions of the set of clauses. A subsumption lattice (Khardon and Arias 2006; Nienhuys-Cheng and Wolf 1997) is such a partial order and has a top node and a bottom node. Upward (generalization) and downward (specialization) refinement operators are defined over clauses as follows. Upward refinement can be achieved by either removing one non redundant literal from the clause or substituting an instantiation with a variable. Downward refinement can be achieved by either adding a non redundant literal to the clause or instantiating a variable with a ground object. A greatest specialization under subsumption (gss) of some finite set S of clauses is defined as the union of all clauses in S after they are standardized apart. Least generalization under subsumption (lgg) of a finite set of clauses S is the clause that is the least general clause reachable from all the clauses in S by repeated upward refinements. Search in the lattice can be top down (using gss) or bottom up (using lgg) refinements to find the best clause.

An implication ordering is a generality ordering based on the relation $\models$ over a set of clauses. Implication ordering in horn clauses is undecidable. Implication order is less tractable and complicated in implementation than subsumption order. However, ability to deal with recursive clauses efficiently and to construct least generalizations efficiently makes implication ordering interesting.

Incorporating background knowledge in the learning model helps ILP systems to explore the lattice in an efficient way. At each node, the current theory combined with the background knowledge is used to compute the scores and thus plays an important role in pruning away a significant portion of the search space. However, as stated before, the heuristic is mostly based on logical coverage of examples and cannot guarantee an optimum model. Since the search space is super exponential, an exhaustive search for optimum model is infeasible in real world settings. Interested readers may please refer to (Nienhuys-Cheng and Wolf 1997) for more details on ILP systems. In the next section, we discuss the challenges involved in extending the StructRELHKL, that learns optimum model in propositional activity recognition settings, to first order settings to find an optimum first order model.

4 Learning Optimum Features in First Order Settings

RELHKL and StructRELHKL, as discussed in Section 2, deals with propositional binary features and the learning involves selection of optimum features from a lattice that can be a conjunction or disjunction lattice over propositional features. The kernels are (or can be decomposed into) binary kernels. The approach starts with an empty set of features and keeps on adding sufficiency condition violating nodes to find the optimum model. Since, one has to deal with a huge set of possible groundings of clauses as well as variable sharing among predicates in clauses of a first order setting, leveraging RELHKL approaches to learn optimal first order relations is non trivial and is an open question. Since subsumption ordering is more tractable and easily implementable, in this section, we use subsumption lattice as the default ordering unless stated otherwise. In the following paragraphs, we look into the problem from an RELHKL perspective and then discuss the challenges involved in mapping the first order setting into RELHKL setting.

RELHKLL or StructRELHKL setting selects optimum features from lattice of propositional binary features, where each node represents a conjunction (or disjunction) of basic features. In the case of activity recognition, each binary sensor value is a basic feature. StructRELHKL considers separate lattices for each activity and selects activity specific features. It also adds all the possible transition features with the set of emission features to get the final model. The hierarchical regularizer imposed over the lattice of features selects a sparse and optimum set of simple features. It also involves summation of binary values over descendants, which is handled efficiently by a kernel trick on basic binary features. Unlike in ILP settings, RELHKL explores multiple nodes simultaneously and thus gives an optimum solution in an efficient way. In the rest of the section, we use the general term RELHKL for RELHKL as well as StructRELHKL.

By extending RELHKL for first order settings, we expect the features (and kernels) to be represented as binary features which can be ordered as a lattice. We confine our discussion to disjunction lattice in the rest of the section. In a disjunction lattice, there should be one node for each possible first order rule (clause), which is a disjunction of predicates with arguments that are variables or ground objects. One way to find a general clause in subsumption lattice is by replacing ground objects in predicates with variables. Following the generalization principle, least general generalizations for a set of clauses are constructed by finding the least general clause that is a generalization of all the individual clauses in the set. Thus a general clause with a variable can have child clauses which have same predicate names but different ground objects in place of the variable. Since there are a huge number of possible groundings for a predicate, it is highly computational to explicitly represent all the possible groundings in a learning setting. For example *breakFastUtensil(X)* can be a generalization for the predicates such as *breakFastUtensil(cup1)*, *breakFastUtensil(cup2)*,..., *breakFastUtensil(cup24)*, *breakFastUtensil(plate1)*, *breakFastUtensil(plate2)*,..., *breakFastUtensil(plate14)*, *breakFastUtensil(bowl1)*, *etc.* Representing every ground predicate, and therefore clause, in the lattice is infeasible in large settings. Another generalization operation is by removing a literal from the clause. For example, *boiling(X) :- hot(X)* is an lgg of *boiling(X) :- hot(X), stoveOn(X)* and *boiling(X) :- hot(X), inductionCooker(X)*. Since RELHKL is not a guided search method to find best clauses, we need to define methods to incorporate all such information in the training model. We now list a few open questions to be answered to build an RELHKL model for first order clauses.

The key questions to be answered for defining an RELHKL setting for first order clauses are a) how to incorporate the huge set of groundings?, b) how to define variable sharing across predicates?, c) how to represent subsumption relationships among clauses in binary kernels d) how to deal with clauses that are subsume equivalent?, e) how to apply refinement operators in the lattice statically or on-the-fly? f) how to incorporate background knowledge?, We briefly discuss these questions in the following paragraphs.

First order systems learn general clauses to represent relationships among a large set of ground objects in the form of conjunctions/disjunctions, where as RELHKL assumes features in the propositional space and learns conjunctions/disjunctions of them. Therefore, the two approaches differ in their representation and lattice setting. To learn optimum first order models using RELHKL, we need to either find a mechanism to represent first order lattice in an RELHKL setting or adapt RELHKL to work on first order settings. One quick fix to this is to assume each first order predicate as a basic propositional feature and learn conjunctions/disjunctions on them. This assumption, in most of the cases, results in information loss as the predicates in a first order clause usually shares variables. On the other hand, grounding all the clauses to learn feature conjunction/disjunctions will lead to a super exponential learning setting and is infeasible in real world settings. Therefore, we need a mechanism to define variable sharing across predicates in a clause in the RELHKL setting for first order clauses.

Popular ILP systems search for useful clauses in the subsumption lattice and any optimal learning approach needs to explore a subsumption lattice (or an implication order) using refinement operators to learn efficiently. In first order logic, two clauses are considered to be subsume equivalent if they are variants after reduction by a renaming scheme θ (Nienhuys-Cheng and Wolf 1997). First order learning systems should not generate subsume equivalent clauses while doing upward and downward refinement operations. The first order optimum feature learning method should be modeled to differentiate between subsume equivalent clauses and the refined clauses.

One of the greatest advantages of first order settings is the ability to incorporate background knowledge. Theories are learned by constructing clauses that when combined with background knowledge gives an explanation to the examples. As background knowledge plays an important role in the efficiency of the trained model, RELHKL for first order settings should be able to incorporate background knowledge in the learning setting.

5 Conclusion and Future work

Feature induction techniques based on heuristics have gained profound interest in the past due to the performance improvement over traditional systems in sequence classification problems in activity recognition domain. These approaches neither guarantee an optimum model nor scales well for an exhaustive search. Recent work on Rule Ensemble Learning using Hierarchical kernels in structured output spaces (StructRELHKL) have been shown to search the lattice for an optimum model in an efficient way and outperforms existing approaches. However StructRELHKL cannot be trivially extended to first order settings. In this paper, we bring the intuitions to leverage StructRELHKL to learn optimum model for first order settings. We also have discussed some of the challenges involved in this domain.

Future work involves developing representation formalisms to incorporate complex first order logic properties such as variable sharing, subsumption relation, background knowledge, *etc.* in the StructRELHKL setup.

Glossary

Atoms: They are predicates in pure form, for *eg: parent(ann,mary)*, *female(X)*.

Body: Right side of (:-) (`if`) is called body of the clause.

Clause: Disjunction of literals for *eg:* (parent(ann,mary) $\vee \neg$ female(ann)) $\equiv$ (parent(ann,mary) :- female(ann)).

Clause Representation: Any clause can be written in the form of

`Comma separated positive literals` :- `Comma separated negated literals`,

where (:-) is pronounced as `if`. For example in propositional logic the clause $a \vee b \vee \neg c \equiv a,b :- c$.

Conjunctive Normal Form (CNF)**:** Every formulae in propositional logic or first-order logic is equivalent to a formula that can be written as a conjunction of disjunctions i.e, something like $(a(X) \vee b(X)) \wedge (c(X) \vee d(X)) \wedge \cdots$. When written in this way the formula is said to be in conjunctive normal form or CNF.

Constants: A constant symbol represents an individual in the world. In first order logic it is represented by small letter *eg: jane*, *1*, *a etc.*

Definite Clause: Clauses with exactly one positive literal *eg: p(X) :- c(X),d(Y).*

Facts: Body less horn clauses, for *eg: female(ann)*; *daughter(mary).*

Functions: Take input as tuple of objects and return another object *eg: motherOf(ann)*, *parentOf(mary).*

Ground Clause: Clauses formed as a result of replacing each variable by all possible constants in each predicate of a clause.

Head: Left side of (:-) (*if*) is called head of the clause.

Herbrand Interpretation: A (Herbrand) interpretation is a truth assignment to all the atoms formed as a result of replacing the variables in a predicate by all the possible constants (objects).

Herbrand Model: a Herbrand model is simply a Herbrand interpretation that makes a wellformed formula true.

Horn Clause: Clause with atmost one positive literal for *eg: (:- parent(ann,joan), female(joan).)* , *(parent(ann,kan) :- female(mary).)*

In a horn clause in CNF, all the atoms preceded by a ¬ form the body part and the atom not preceded by a ¬ is the head. Here ¬A means (not)A.

Knowledge Base: A Knowledge Base is a set of clauses which represents a theory.

Literals: They are predicates in either pure form or negated form, for *eg: ¬parent(ann,mary)*.

Model: An interpretation which makes the clause true. For eg: P :- Q, R, the models are $M = \phi, \{P, Q, R\}$.

Statistical Relational Learning (SRL): Statistical relational learning deals with machine learning and data mining in relational domains where observations may be missing, partially observed, and/or noisy.

Variables: Stands for typed objects and starts with the capital letters for *eg: X,Abs*, *etc.*

References

De Raedt, L. 1997. Logical settings for concept-learning. *Artif. Intell.* 95(1):187–201.

Domingos, P.; Kok, S.; Poon, H.; Richardson, M.; and Singla, P. 2006. Unifying logical and statistical ai. *AAAI*.

Forney, G.D., J. 1973. The viterbi algorithm. *Proceedings of IEEE* 61(3):268–278.

Gibson, C.; van Kasteren, T.; and Krose, B. 2008. Monitoring homes with wireless sensor networks. *Proceedings of the International Med-e-Tel Conference*.

Jawanpuria, P.; Jagarlapudi, S. N.; and Ramakrishnan, G. 2011. Efficient rule ensemble learning using hierarchical kernels. *International Conference on Machine Learning*.

Kersting, K., and De Raedt, L. 2001. Bayesian logic programs. Technical report.

Kersting, K.; Raedt, L. D.; and Raiko, T. 2006. Logical hidden markov models. *Journal of Artificial Intelligence Research* 25.

Khardon, R., and Arias, M. 2006. The subsumption lattice and query learning. *J. Comput. Syst. Sci.* 72(1):72–94.

Lafferty, J.; McCallum, A.; and Pereira, F. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. ICML.

Landwehr, N.; Gutmann, B.; Thon, I.; Raedt, L. D.; and Philipose, M. 2009. Relational transformation-based tagging for activity recognition. *Fundamenta Informaticae, v.89 n.1, p.111-129*.

McCallum, A. K. 2003. Efficiently inducing features of conditional random fields. Proceedings of the Nineteenth Conference Annual Conference on Uncertainty in Artificial Intelligence.

Muggleton, S., and Feng, C. 1990. Efficient induction of logic programs. In *New Generation Computing*. Academic Press.

Muggleton, S. 1995. Inverse entailment and progol. *New Generation Computing* 13:245–286.

Nair, N.; Saha, A.; Ramakrishnan, G.; and Krishnaswamy, S. 2012. Rule ensemble learning using hierarchical kernels in structured output spaces. *AAAI 2012*.

Nair, N.; Ramakrishnan, G.; and Krishnaswamy, S. 2011. Enhancing activity recognition in smart homes using feature induction. *International Conference on Data Warehousing and Knowledge Discovery*.

Natarajan, S.; Bui, H. H.; Tadepalli, P.; Kersting, K.; and keen Wong, W. 2008. Logical hierarchical hidden markov models for modeling user activities. In *In Proc. of ILP-08*.

Nienhuys-Cheng, S.-H., and Wolf, R. d. 1997. *Foundations of Inductive Logic Programming*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Quinlan, J. R., and Cameron-Jones, R. M. 1993. Foil: A midterm report. *European Conference on Machine Learning*.

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

Raedt, L. D., and Dzeroski, S. 1994. First order jk-clausal theories are pac-learnable. *Artificial Intelligence* 70:375–392.

Raedt, L. D.; Frasconi, P.; Kersting, K.; and Muggleton, S. 2008. *Probabilistic Inductive Logic Programming*, volume 8.

Richardson, M., and Domingos, P. 2006. Markov logic networks. *Mach. Learn.* 62(1-2):107–136.

Srinivasan, A. 2007. The aleph manual. *Technical Report, University of Oxford*.

Tsochantaridis, I.; Hofmann, T.; Joachims, T.; and Altun, Y. 2004. Support vector machine learning for interdependent and structured output spaces. *International Conference on Machine Learning*.

Tsochantaridis, I. 2006. Support vector machine learning for interdependent and structured output spaces.

van Kasteren, T.; Noulas, A.; Englebienne, G.; and krose, B. 2008. Accurate activity recognition in a home setting. *10th International conference on Ubiquitous computing*.

Wilson, D. H. 2005. Assistive intelligent environments for automatic health monitoring. *PhD Thesis, Carnegie Mellon University*.