Late-Breaking Developments in the Field of Artificial Intelligence
Papers Presented at the Twenty-Seventh AAAI Conference on Artificial Intelligence

Learning CP-net Preferences Online from User Queries*

Joshua T. Guerin
The University of Tennessee at Martin
110 Business Administration Building
Martin, Tennessee 38238
jguerin@utm.edu

Thomas E. Allen
University of Kentucky
Department of Computer Science
Davis Marksbury Building
329 Rose Street
Lexington, Kentucky 40506-0633
thomas.allen@uky.edu

Judy Goldsmith
University of Kentucky
Department of Computer Science
311 Davis Marksbury Building
329 Rose Street
Lexington, Kentucky 40506-0633
goldsmit@cs.uky.edu

Introduction

CP-nets (Boutilier et al. 1999) offer a compact qualitative representation of human preferences that operate under *ceteris paribus* ("with all else being equal") semantics. In this paper we present a novel algorithm through which an agent learns the preferences of a user. CP-nets are used to represent such preferences and are learned online through a series of queries generated by the algorithm. Our algorithm builds a CP-net for the user by creating nodes and initializing CPTs, then gradually adding edges and forming more complex CPTs consistent with responses to queries until a confidence parameter is reached. Our algorithm does not always converge to the original CP-net, but our experiments show that it can learn a CP-net that closely tracks with the original for a series of outcome comparison queries. Our work builds upon previous CP-net learning research, particularly that of (Lang and Mengin 2008; 2009) and (Dimopoulos, Michael, and Athienitou 2009). Other CP-net learning algorithms include (Eckhardt and Vojtáš 2009; 2010), (Eckhardt and Vojtáš 2009; 2010), (Koriche and Zanuttini 2009; 2010), and (Liu et al. 2012). Our algorithm differs in that in is guaranteed to produce a CP-net in polynomial time given a constant bound on the number of parents.

Modeling Preferences with CP-nets

By *preference*, we mean a strict partial order $\succ$ over a set of *outcomes* $\mathcal{O}$. Such outcomes can be factored into *variables* $\mathcal{V}$ with associated (binary) *domains* $\mathrm{Dom}(\mathcal{V})$: $\mathcal{O} = v_1 \times v_2 \times \cdots \times v_k$. We define $o[i]$ as the *projection* of outcome o onto variable v_i. Note that the number of outcomes and orderings is exponential in the number of variables. *Conditional Preference networks* (CP-nets) generally offer a more compact representation.

Definition 1. *A* CP-net $\mathcal{N}$ *is a directed graph. Each node* v_i *represents a preference over a finite domain. An edge* (v_i, v_j) *indicates that the preference over* v_j *depends on* v_i*. If a node has no incoming edges, the preference involving its variable is not conditioned on other variables. A* conditional preference table (CPT) *is associated with each node* v *and specifies the preference over* $\mathrm{Dom}(v)$ *as a function of the values assigned to its* parent nodes $\mathrm{Pa}(v)$*. A* separable CP-net *is one with no edges—no variable depends on any other.*

To guarantee tractability, we make some simplifying assumptions: 1. Cycles are disallowed. 2. We restrict to binary domains. 3. A maximum bound p is placed on the number of parents a node may have: We conjecture that most human preferences are conditioned on 3–5 nodes and thus feel justified in assuming such a bound.

Algorithm

Our algorithm consists of two phases. First, it constructs a separable CP-net with default CPTs. Next, it successively attempts to refine the model, adding edges and learning more complex CPTs consistent with evidence drawn from the *user queries.* (See LEARN-CP-NET and its subroutine FIND-PARENTS [Alg. 1 and 2]).

Phase 1 constructs a *separable CP-net basis* by asking the user to provide a default preference for each $v_i \in \mathcal{V}$.

Definition 2. *Let* v_i *be a variable in a CP-net with binary domain* $\mathrm{Dom}(v_i) = \{x_i, y_i\}$*. An* attribute comparison query *is one in which we present the user the values* x_i *and* y_i *and ask whether* $x_i \succ y_i$ *or* $y_i \succ x_i$*.*

The result is a CP-net with no edges and only the default values. However, we are *unconfident* that all preferences are unconditional. Here we model *confidence* q as a parameter in our algorithm, defining disjoint sets CONFIDENT and UNCONFIDENT s.t. $v \in$ CONFIDENT iff we are confident that the preferences over v are conditioned only by its parent variables in the graph of $\mathcal{N}$.

In the second phase we refine $\mathcal{N}$ by discovering such conditional relationships as may exist between variables by asking the user's preference over pairs of outcomes.

Definition 3. *In an* outcome comparison query*, we provide the user a pair of outcomes,* $\{o_1, o_2\} \in \mathcal{O}$*. The user responds with* $o_1 \succ o_2$*,* $o_2 \succ o_1$ *or* $o_1 \sim o_2$*, indicating that the user strictly prefers the first outcome to the second, the second to the first, or is indifferent.*

Definition 4. *A* random query *is an outcome comparison query in which all values of* o_1 *and* o_2 *are selected uniformly randomly from their domains, with the requirement that the query must be* relevant *to node* v_i*: that is,* $o_1[i] \neq o_2[i]$*. A*

*This material is based upon work supported by the National Science Foundation under Grants No. CCF-1215985 and CCF-1049360. The content reflects the views of the authors and not necessarily those of NSF.

Algorithm 1 LEARN-CP-NET($\mathcal{V}$, p, q)

$\mathcal{N} \leftarrow \emptyset$; *comparisons* $\leftarrow \emptyset$
confident $\leftarrow \emptyset$; *unconfident* $\leftarrow \mathcal{V}$
for $v_i \in \mathcal{V}$ **do**
  query user: do you prefer $x_i \succ y_i$ or $y_i \succ x_i$?
  v_i.CPT $\leftarrow$ default CPT based on user response
  insert v_i into $\mathcal{N}$
end for
repeat
  for $r \leftarrow 0$ **to** p **do**
    for $v_i \in$ *unconfident* **do**
      $(P, C) \leftarrow$ FIND-PARENTS(v_i, r, q)
      if $C \neq$ FAIL **then**
        v_i.CPT $\leftarrow C$
        add edges from all P to v_i
        move v_i from *unconfident* to *confident*
      end if
    end for
  end for
until no parents added this iteration
return $\mathcal{N}$

Algorithm 2 FIND-PARENTS(v_i, r, q)

for $P \in$ {all subsets of *confident* of size r} **do**
  $(C,$ *evidCount*$) \leftarrow$ CREATE-CPT(v_i, P)
  while $(C \neq$ FAIL**)** **and** (*evidCount* $< q$) **do**
    $(o_1, o_2) \leftarrow$ generate random query for v_i
    query user: do you prefer outcome o_1 or o_2?
    add o_1, o_2 to *comparisons* in specified order
    $(C,$ *evidCount*$) \leftarrow$ CREATE-CPT(v_i, P)
  end while
  if $C \neq$ FAIL, **return** (C, P), **end if**
end for
return (FAIL, $\emptyset$)

random adaptive query *adds the additional requirement that for all* $v_j \in$ CONFIDENT, $o_1[j] = o_2[j]$.

Random adaptive queries provide a heuristic that may reduce the search space for a CP-net by not continuing to analyze nodes once they are labeled CONFIDENT.

We search first for nodes that do not need parents. For each node $v \in \mathcal{V}$, we ask a series of outcome comparison queries, then iterate over orderings provided by the user and stored in COMPARISONS. If the user prefers $x_i \succ y_i$ or $y_i \succ x_i$ in *all instances*, we conclude that the preferences over v_i are *unconditional* and move it from UNCONFIDENT to CONFIDENT. If we have not accumulated enough evidence, we continue querying the user. While UNCONFIDENT $\neq \emptyset$, we continue trying to refine our model with new conditional relationships, represented as edges and more complex CPTs. For each unconfident node, we iterate over potential sets of parent nodes of increasing size up to p. If, in an iteration, we fail to add parents for any nodes, we stop. Our algorithm will always output a CP-net, possibly with some CPTs in their default state from Phase 1; however, this rarely occurred in our tests, and only in overtrained CP-nets of minimal size.

For a given target node and set of possible parents, we construct a 2-SAT instance such that (1) a satisfying assignment tells us that the target node's values are consistent with the given set of parents and (2) the assignment to variables gives us the entries of the target node's CPT. Our method for this closely follows (Dimopoulos, Michael, and Athienitou 2009), to which the reader is referred for specifics.

Analysis and Experiments

Theorem 1. LEARN-CP-NET *is* resolute*—i.e., it is guaranteed to output a consistent CP-net* $\mathcal{N}$*—and runs in time polynomial in* n^p *and* q *in the worst case.* (Proof omitted.)

We generated random CP-nets for given n and p, and used them to generate responses to the queries for our learning algorithms. We looked at computation time (as a function of n, p, and q), and the accuracy of the learned CP-net, i.e., on how many possible comparison queries do the generated and learned CP-nets agree. The tables below show the metrics of the learned CP-net $\mathcal{N}_L$ compared with the training model $\mathcal{N}_T$ over a series of experiments.[1] We set $p = 5$; since we used $\delta = n$, most nodes didn't have p parents. However, in trials with $\delta = cn$ for $c = 2, 3, \ldots$, we saw very similar graphs. The metrics shown are averages over 10 trials. Table 1 shows that agreement was generally 75–90%+ with the proper q. As shown, disagreement between models was rare, but as n increases, the learned model is more likely to be indecisive about preferences on which the training model decides. Increasing q sometimes has an *adverse* effect on the agreement the models; if q is too high, the model can be *overtrained*. We also found that for some q values, computational time did not grow monotonically. When we generate queries to learn the CPT for v_i, those queries may be relevant to other nodes in UNCONFIDENT. It may be that, when we come to v_j, we already have q many relevant comparisons.

Table 1: Agreement of $\mathcal{N}_L$ with $\mathcal{N}_T$

q	$n = 3$	$n = 5$	$n = 7$	$n = 10$
4	0.9964	0.7996	0.6086	0.5118
6	0.9964	0.9221	0.7653	0.7677
8	1.0000	0.9667	0.8918	0.6990
10	0.9964	0.9735	0.8583	0.6760
12	0.9964	0.9621	0.9171	0.6597
14	1.0000	0.9816	0.8512	0.5518
16	0.9929	0.9434	0.9062	0.5539
18	1.0000	0.9646	0.9237	0.4367
20	0.9964	0.9731	0.8395	0.5886

Table 2: Disagreement of $\mathcal{N}_L$ with $\mathcal{N}_T$

q	$n = 3$	$n = 5$	$n = 7$	$n = 10$
4	0.0036	0.0696	0.0880	0.0426
6	0.0036	0.0430	0.0545	0.0389
8	0	0.0238	0.0375	0.0280
10	0.0036	0.0216	0.0257	0.0254
12	0.0036	0.0174	0.0188	0.0150
14	0	0.0184	0.0259	0.0118
16	0.0071	0.0309	0.0250	0.0127
18	0	0.0228	0.0186	0.0095
20	0.0036	0.0269	0.0236	0.0110

[1] Due to space constraints, indecision results have been omitted; these values can be computed from Tables 1–2.

References

Boutilier, C.; Brafman, R. I.; Hoos, H. H.; and Poole, D. 1999. Reasoning with conditional ceteris paribus preference statements. In *UAI-99*, 71–80.

Dimopoulos, Y.; Michael, L.; and Athienitou, F. 2009. Ceteris paribus preference elicitation with predictive guarantees. In *IJCAI-09*, 1890–1895. San Francisco, CA, USA: Morgan Kaufmann.

Eckhardt, A., and Vojtáš, P. 2009. How to learn fuzzy user preferences with variable objectives. In *IFSA/EUSFLAT*, 938–943.

Eckhardt, A., and Vojtáš, P. 2010. Learning user preferences for 2CP-regression for a recommender system. In *SOFSEM-10*, 346–357.

Koriche, F., and Zanuttini, B. 2009. Learning conditional preference networks with queries. In *IJCAI-09*, 1930–1935.

Koriche, F., and Zanuttini, B. 2010. Learning conditional preference networks. *Artificial Intelligence* 174:685–703.

Lang, J., and Mengin, J. 2008. Learning preference relations over combinatorial domains. In *NMR-08*.

Lang, J., and Mengin, J. 2009. The complexity of learning separable ceteris paribus preferences. In *IJCAI-09*, 848–853. San Francisco, CA, USA: Morgan Kaufmann.

Liu, J.; Xiong, Y.; Wu, C.; Yao, Z.; and Liu, W. 2012. Learning conditional preference networks from inconsistent examples. *TKDE* PP(99):1.