Discriminative Non-Parametric Learning of Arithmetic Circuits

Nandini Ramanan [1] **Mayukh Das** [1] **Kristian Kersting** [2] **Sriraam Natarajan** [1]

Abstract

Arithmetic Circuits (AC) and Sum-Product Networks (SPN) have recently gained significant interest by virtue of being tractable deep probabilistic models. Most previous work on learning AC structures, however, hinges on inducing a tree-structured AC and, hence, may potentially break loops that may exist in the true generative model. To repair such broken loops, we propose a gradient-boosted method for structure learning of discriminative ACs (DACs), called DACBOOST. Since, in discrete domains, ACs are essentially equivalent to mixtures of trees, DACBOOST decomposes a large AC into smaller tree-structured ACs and learns them in a sequential, additive manner. The resulting non-parametric manner of learning the DACs results in a model with very few tuning parameters making our learned model significantly more efficient. We demonstrate on standard data sets and some real-world data sets, the efficiency of DACBOOST compared to the state-of-the-art DAC learners without sacrificing the effectiveness. This makes it possible to employ DACs for large scale real-world tasks.

1. Introduction

The application and adaptation of probabilistic graphical models for real problems such as bio-medicine, computational biology have tremendously increased in the recent years. Particularly, there is a surge of interest in tractable probabilistic models where inference is significantly more efficient (Zhao et al., 2016; Darwiche, 2003; Poon & Domingos, 2011). Of these models, both Arithmetic Circuits (AC) (Darwiche, 2003) and Sum-Product Networks (SPN) (Poon & Domingos, 2011) have garnered particular interest due to their mutual equivalence and their ability to model several other tractable models (Rooshenas & Lowd, 2016).

As pointed out by Rooshenas and Lowd (Rooshenas & Lowd, 2016), most of the learning methods developed for these models are generative. In the discriminative case, as they argue, ACs are a better fit for better capturing log-linear models due to their ability to directly represent the parameters in their nodes (as against weights in SPNs). Consequently, they developed a discriminative learning method for ACs that greedily searches through the space of features. While successful, their work has two limitations: (1) a large number of parameters that must be tuned and (2) the ACs are typically limited to being tree-structured and, hence, may break loops.

To overcome these limitations, we propose the first non-parametric learning method for discriminative ACs (DACs) based on gradient-boosting, called DACBOOST. Inspired by the intuition that many weak learners, in our case tree-structured ACs, could be more successful in learning a conditional distribution, DACBOOST introduces parameters as necessary. We derive the gradient updates that are used to reweigh the examples after each iteration and present the algorithm for learning weak, tree-structured ACs in a sequential manner. The benefits of DACBOOST are two-fold. First, it can repair broken loops by mixing different tree-structured ACs in a stage-wise manner. Second, it reduces the space of structure search and parameter updates at the same time thus avoiding the seemingly difficult task of repeated full parameter estimation when scoring each structure. In our extensive experiments on both the standard data sets due to Rooshenas and Lowd (Rooshenas & Lowd, 2016) and on five real-world data sets, we demonstrate both the effectiveness and efficiency of this boosted DAC approach.

Overall, this paper makes a number of important contributions. We present the first ensemble based learning approach for ACs. Establishing this link is especially significant because (as we state later) structure learning of complete and valid SPNs or ACs — for discrete domains, SPNs and ACs are equivalent (Rooshenas & Lowd, 2016) — is difficult (Zhao et al., 2016). Boosting reduces the space of possible structure search for learning the structure (the weak ACs) and parameter (leaves of these ACs) simultaneously, hence rendering the learning task more practical. Second, most of the prior approaches have focused on tree-

[1]University of Texas at Dallas, USA [2]TU Darmstadt, Germany. Correspondence to: Nandini Ramanan <nandini.ramanan@utdallas.edu>.

Proceedings of the Workshop on Tractable Probabilistic Models (TPM) at ICML 2019.

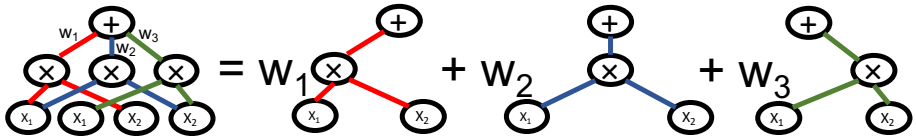
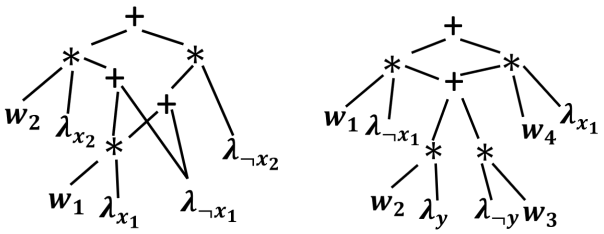


Figure 1. Illustration due to (Zhao et al., 2016) that a complete and decomposable AC, here actually an SPN, is a mixture of trees. The leaves indicate univariate distributions over X_1 and X_2. Different colors are used to highlight unique trees, which are products of univariate distributions. (Best viewed in color)

structured ACs to retain tractability in learning. Triggered by the view of ACs as mixtures of trees (Zhao et al., 2016), DACBOOST extends these tree-structured learners towards learning valid and complete ACs via boosting. Third, our experimental evaluations on both standard and some novel and interesting real-world data clearly establish the superiority of our learner. In nearly all the domains, we achieve equal or better performance for a fraction of the learning time when compared to the state-of-the-art DAC learner.

We proceed as follows. After reviewing the necessary background, namely ACs and boosting conditional distributions, we introduce and discuss DACBOOST. Before concluding, we provide empirical evidence on seven standard benchmark and five novel data sets.



(a) Arithmetic Circuit (AC) (b) Conditional AC

Figure 2. **Left:** Example arithmetic circuit that represents a markov random field over two variables x_1 and x_2 and having potentials w_1 and w_2 for the 2 features $x_1 \wedge x_2$ and x_2. **Right:** Conditional AC that represents the distribution $P(y|x_1)$ where y is a query variable and x_1 is an evidence variable.

2. Background

Let us start off by reviewing the work on arithmetic circuits and functional gradient boosting.

2.1. Arithmetic Circuits

Probabilistic graphical models have long been successfully in a wide variety of modeling tasks. There is an increased interest in tractable probabilistic models since in traditional models inference is a function of the tree-width and hence complex (Chandrasekaran et al., 2008). There is a necessity for tractability in the presence of large amounts of evidence. Consequently, there has been significant progress in tractable probabilistic modeling in several directions: (1) bounded tree-width models (Bouman & Shapiro, 1994; Bach & Jordan, 2002; Gogate et al., 2010; Karger & Srebro, 2001), (2) probabilistic models with tractable factors such as Markov Random Fields with sub-modular potentials (Osokin et al., 2011; Vernaza et al., 2008), (3) structure compression by exploiting share-able parameters such as associative Markov Networks (Taskar et al., 2004; Munoz et al., 2008) and (4) compiling models into representations suitable for efficient inference such as (deep) probabilistic architectures like Arithmetic Circuits (ACs) (Darwiche, 2003; Lowd & Domingos, 2008; Lowd & Rooshenas, 2013) and Sum-Product Networks (SPNs) (Poon & Domingos, 2011; Gens & Domingos, 2013; Rooshenas & Lowd, 2014). We consider ACs and explore an efficient learning algorithm while retaining the tractability.

The probability distribution induced by a probabilistic graphical model can be represented by a multilinear function called a *network polynomial* that consists of sum of product of indicator variables λ and the parameters w of the network (Darwiche, 2003). This enables answering marginal and conditional queries in linear time in the size of the polynomial by setting the λ consistent with instantiations. However, network polynomials have exponential number of terms one for each possible state of the set of random variables, possibly making inference intractable. In these cases, however, ACs can be used to compactly represent the network polynomial.

An *Arithmetic Circuit* $AC(\mathcal{X})$ is a rooted, directed acyclic graph over the variables $\mathcal{X}$. It contains $+$ or $*$ as internal nodes and its leaf nodes are labeled with either a non negative parameter w or an indicator variable λ. Then for any instantiation x, the value of the circuit $AC(\mathcal{X})$ is computed by assigning indicator λ_x the value 1 if $\mathcal{X}$ is compatible with instantiation x and 0 otherwise. For example, consider a simple AC as shown in Fig. 2(a) over two binary random variables x_1 and x_2. Note that by the original definition, ACs are generative and hence model a joint distribution. The network polynomial for the above AC, which is multilinear

in the λ and w variables, can be written as;

$$\lambda_{x_1}\lambda_{x_2}w_1w_2 + \lambda_{x_1}\lambda_{\neg_{x_2}} + \lambda_{\neg_{x_1}}\lambda_{x_2}w_2 + \lambda_{\neg_{x_1}}\lambda_{\neg_{x_2}}$$

The complexity of evaluating an AC is linear in the size of the circuit. The two properties that lead to tractable ACs as discussed in (Darwiche, 2003; Lowd & Rooshenas, 2013) are, (1) *Decomposibility* : An AC is decomposable *iff* $\forall$ pairs of nodes $\langle l_1, l_2\rangle$: l_1 and l_2 are children of $*$ node (product) $\Rightarrow var(l_1) \wedge var(l_2) = \emptyset$ (2) *smoothness* : An AC is smooth *iff* $\forall$ nodes : l is a child of $+$ node n and $var(l) = var(n)$. Zhao *et al.* (Zhao et al., 2016) have shown that ACs can be viewed as mixtures of trees as illustrated in Fig. 1.

2.2. Learning (Conditional) Arithmetic Circuits

Given that a valid AC can efficiently marginalize over any variable, they have attracted significant attention. Lowd and Rooshenas (Lowd & Rooshenas, 2013) proposed an algorithm to learn circuits directly from data instead of just compiling them from models. Compiling a Bayesian Network or Markov Network to a valid AC is an expensive task as pointed out by Rooshenas and Lowd (Rooshenas & Lowd, 2016) due to exponential blow up in the size of the network. However, when learning discriminatively, *i.e.* $P(\mathcal{Y}|\mathcal{X})$, one does not have to marginalize over the variables that appear in evidence $\mathcal{X}$. Rooshenas and Lowd suggest that replacing the indicators λ corresponding to the evidence variables with a constant in a (Conditional) discriminative AC (DAC) does not impact the smoothness or decomposability over the query variables (Fig. 2(b) illustrates a Conditional AC with one query and one evidence variable).

In most prediction tasks, however, we are interested in computing the conditional distribution of a pre-determined set of response variables given their parents. Thus, modeling such a conditional via a DAC, where we can treat the entire set of parents as evidence variables, is sufficient. This allows us to exploit the advantages of discriminative training of DACs, as pointed out earlier, for learning tractable probabilistic models from labeled data. Rooshenas and Lowd (Rooshenas & Lowd, 2016) proposed DACLEARN and showed how it can learn the structure of a DAC by optimizing the penalized conditional log-likelihood (CLL). Their work provides an insight into how optimizing the CLL ($\log P(\mathcal{Y}|\mathcal{X})$, where $\mathcal{X}$ is the set of evidence variables) for learning DAC, is considerably more tractable even for large tree-width models, as opposed to generative training of a typical AC. DACLearn, thus, iteratively grows the DAC, choosing a set of most informative features at each step of the gradient of penalized CLL. Their structure update technique follows from their previous work on transforming MRFs to ACs (Lowd & Rooshenas, 2015; 2013).

A similar approach has been proposed for discriminative learning of SPNs by Gens and Domingos (Gens & Domingos, 2012; 2013), an equivalent representation based on network polynomials. Note that the same insights and properties of conditional validity and smoothness as DACs apply to discriminative SPNs, when learning conditional distributions over query variables given the evidence. Hence, this work also optimizes penalized conditional log-likelihood. Gens and Domingos, however, analyze the effects of 2 different types of inference, namely, marginal inference and Most Probably Explanation (MPE), for structure scoring and parameter estimation during learning. They show that hard inference via MPE results in better learning as it alleviates the gradient diffusion problem in a deep probabilistic model such as SPN.

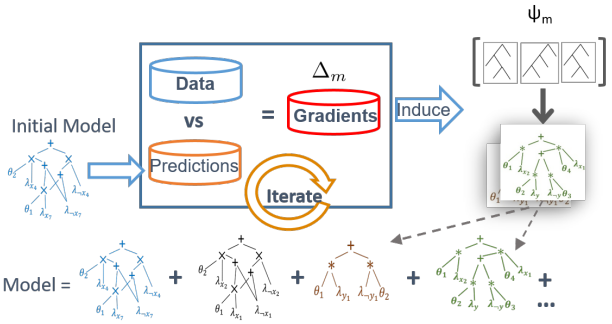


Figure 3. Learning of Discriminative Arithmetic Circuits via boosting. As can be seen, at each iteration a small weak DAC is learned for each query variable. Once a DAC is learned at each iteration, the weights of the examples are computed and a new AC is learned. These are then added to the model and the process continues until convergence.

2.3. Gradient Boosting for Conditionals

Functional Gradient Boosting (FGB) is motivated by the intuition that inducing an ensemble of weak models to change one's probabilistic predictions locally can potentially be more expressive and efficient than finding a single, highly accurate model. Specifically, the problem of learning conditional probabilistic models is transformed into learning a sequence of function approximation problems (following the work by Friedman (Friedman, 2001)). Thus a conditional probability distribution is represented as a weighted sum of regression models learned sequentially via a stage-wise optimization (Natarajan et al., 2012; Khot et al., 2011).

For a particular example y_i its conditional distribution given its parents $\mathbf{x}_i$ can be learned by fitting a model $P(y|\mathbf{x}) \propto e^{\psi(y,\mathbf{x})}$, and can be expressed non-parametrically in terms of a potential function $\psi(y_i;\ \mathbf{x}_i)$:

$$P(y_i\,|\mathbf{x}_i) \;=\; \frac{e^{\psi(y_i;\,\mathbf{x}_i)}}{\sum_{y'}\, e^{\psi(y';\,\mathbf{x}_i)}}$$

Instead of learning in the parameter space (P), gradient

is obtained in the functional-space ψ. The key is to successively approximate ψ as a sum of weak learners, which are typically regression trees. Staring from an initial ψ_0 functional gradient ascent iteratively adds gradients Δ_i. For every iteration i a new weak model h_i is fitted to the gradient. After m iterations, the potential is given by $\psi_m = \psi_0 + \Delta_1 + ... + \Delta_m$. Here, Δ_m is the functional gradient at step m,

$$\Delta_m = \eta_m \cdot E_{\mathbf{x},y}\left[\frac{\partial}{\partial \psi_{m-1}} \log P(y \mid \mathbf{x}; \psi_{m-1})\right]$$

where η_m is the learning rate. Dietterich *et al.* (Dietterich et al., 2004) postulated that evaluating the gradient for every training example and fitting a regression tree to these derived examples i.e., fitting a regression tree h_m on the training examples $[(x_i, y_i), \Delta_m(y_i; x_i)]$ is a close and reasonable approximation of the desired Δ_m and essentially, points in the same direction. Thus, ascent in the direction of h_m will approximate the true functional gradient. This approach has been adapted for learning conditional models in conditional random fields (Dietterich et al., 2008), dependency networks (Natarajan et al., 2012), markov networks (Khot et al., 2011), logisitic regression models (Ramanan et al., 2018) and conditional exponential distributions (Yang et al., 2016).

3. Learning Discriminative ACs

Now that we have all the necessary tools to tackle the problem of learning DACs we formally define our problem as:

> **Given** a data set $\mathcal{D}(\mathcal{X}, \mathcal{Y})$, where $\mathcal{Y}$ is a set of query variables, and $\mathcal{X}$ is a set of evidence variables, **find** the structure and parameters of a discriminative arithmetic circuit (DAC), i.e., the distribution $P(\mathcal{Y}|\mathcal{X})$).

We adopt the definition of discrimative ACs (DACs) as described by Rooshenas and Lowd (Rooshenas & Lowd, 2016). A conditional AC defines a tractable conditional probability distribution, i.e., $P(\mathcal{Y} \mid \mathcal{X})$ over the query variables $\mathcal{Y}$ given the evidence $\mathcal{X}$. As specified in their work, the validity of conditional ACs are more relaxed compared to the original definition of ACs. The key difference as pointed out earlier is the fact that one allows for conditioning the query variables over the evidence. This facilitates training DACs in a manner similar to conditional Random Fields (CRFs) (Rooshenas & Lowd, 2016). Note that DACs offer similar benefits as CRFs in modeling complex dependencies between evidence and query while retaining tractability for learning and inference. Given that gradient-boosting is state-of-the-art in learning CRFs (Dietterich et al., 2008; Chen et al., 2015), we derive a learning algorithm based on gradient-boosting for full model learning (structure + parameter learning) of DACs.

3.1. Gradient Boosting for Conditional ACs

Throughout the derivation, we adopt the following convention for notation - subscript i denotes the i^{th} example, superscript j denotes the j^{th} feature of the evidence set and superscript (p) denotes the p^{th} query variable. Given this notation, following DACLearn (Lowd et al. 2016) that optimizes conditional log-likelihood of train data set $\mathcal{D}$ by finding the best set of features f, where, $\mathbf{y} = \mathcal{Y}^{(p)} \subseteq \mathcal{Y}$ and $\mathbf{x} \subseteq \mathcal{X}$, we define $CLL(\mathcal{D}) :=$

$$\sum_{(\mathbf{y},\mathbf{x})\in\mathcal{D}} \log P(\mathbf{y}|\mathbf{x}) = \sum_{(\mathbf{y},\mathbf{x})\in\mathcal{D}} \sum_j w^j f^j - \log Z(\mathbf{x})$$

where Z denotes the normalization constant (partition function).

As mentioned earlier, functional-gradient boosting obtains point-wise gradient for each example separately. So now considering the log-likelihood for a specific example i and a specific query variable y (we drop the superscript for brevity),

$$\begin{aligned}\log P(y_i = 1|\mathbf{x}_i) &= \sum_j w^j f^j(y_i = 1|\mathbf{x}_i) - \log Z(\mathbf{x}_i) \\ &= \sum_j w^j f^j(y_i = 1|\mathbf{x}_i) - \\ &\quad \log \sum_{y'} \exp\big(\sum_j w^j f^j(y_i = y'|\mathbf{x}_i)\big)\end{aligned}$$

To adapt functional gradient boosting to the task of learning DAC, we map this conditional probability definition from the parameter space to the functional space ψ. Like CRFs, the definition of DACs, naturally allow for the functional representation. ψ for i-th example is directly,

$$\psi(y_i|\mathbf{x}_i) = \sum_j w^j f^j$$

Specifically, we denote $\psi(y_i|\mathbf{x}_i)$ as the potential of y_i given $\mathbf{x}_i$. As with CRFs, and other adaptations of gradient-boosting (Natarajan et al., 2012; Khot et al., 2011), we explain the process assuming binary query variables. However, they can be easily extended to the multivalued case. Now, the above equation can be written from the perspective of $y_i = 1$,

$$\begin{aligned}&\log P(y_i = 1|\mathbf{x}_i) = \\ &\qquad \psi(y_i = 1|\mathbf{x}_i) - \log \sum_{y'} \exp(\psi(y_i = y'|\mathbf{x}_i)))\end{aligned}$$

Now computing the pointwise derivative of CLL w.r.t $\psi\ \forall i$ we get;

$$\begin{aligned}\frac{\partial P(y_i|\mathbf{x}_i)}{\partial \psi(y_i|\mathbf{x}_i)} &= I(y_i|\mathbf{x}_i) - \frac{\exp(\psi(y_i = 1|\mathbf{x}_i))}{\sum_{y'} \exp(\psi(y_i = y'|\mathbf{x}_i))} \\ &= I(y_i|\mathbf{x}_i) - P(y_i = 1|\mathbf{x}_i)\end{aligned}$$

Note that the above expression is similar to the gradients for well-known probabilistic models such as CRFs (Dietterich et al., 2008), dependency networks (Natarajan et al., 2012) and Markov (logic) networks (Khot et al., 2011) to name a few. Thus the weight (gradient) of each example is simply the difference between whether the query variable in the current training example is true according to the data (denoted by indicator function I) and the predicted probability of the query variable being true according to the current model. These gradients are then used in the next iteration of boosting where a weak learner (DAC in our case) is learned to fit these pointwise gradients. The score of the structure using the set of candidate features $\mathcal{F}$ can be rewritten as;

$$Score(\mathcal{F}) = \Delta_{cll}(\mathcal{F})$$

where Δ_{cll} denotes the change in CLL. The goal is to identify the features that maximize this change in CLL.

A key aspect of our learning approach is that, unlike in the approach due to Rooshenas and Lowd (Rooshenas & Lowd, 2016), there is no necessity to introduce an explicit penalty on the number of edges and parameters in the circuits. Learning weak models automatically takes care of regularization by controlling the depth of the learned ACs.

Specifically, given the functional-gradients for each example, we learn a small, tree-structured AC by searching through the space of potential features to add next by minimizing the weighted variance of the conditional distribution according to the current model. Once the AC is constructed, the next step is to estimate the weights w^j at the leaves of the DAC. To estimate them, we maximize the increment in CLL function:

$$\Delta_{cll}(\mathcal{F}) = \sum_{(\mathbf{y},\mathbf{x})\in D} \sum_j w^j \hat{P}(f^j|\mathbf{x}) - \Delta \log Z(\mathbf{x})$$

where $\hat{P}$ is the new empirical probability distribution after introducing the new candidate features. Computing the gradient w.r.t. parameter becomes:

$$\frac{\partial \Delta_{cll}(F)}{\partial w^j} = \sum_{(\mathbf{y},\mathbf{x})\in D} \frac{\exp(w^j) P(f^j|\mathbf{x})}{\exp(\Delta \log Z(\mathbf{x}))}$$

Putting the different pieces together, we derive the DACBOOST algorithm that we describe next.

3.2. The DACBOOST Algorithm

Alg. 1 summarizes our boosting approach. Here DACBOOST() is the primary procedure that learns an ensemble of gradient boosted Arithmetic Circuits for a given query variable $\mathcal{Y}^{(p)}$ and training data $\mathcal{D}$. As explained earlier, $\mathcal{X}$ and $\mathcal{Y}$ are sets of evidence and query variables respectively, and so, a training example $\mathcal{D}_i = \langle \mathcal{X}_i, \mathcal{Y}_i \rangle$. The

Algorithm 1 DACBOOST: Boosted Arithmetic Circuits

Require: $\mathcal{X}, \mathcal{Y}, \mathcal{D}, p$
1: Initialize to uniform prior: Model $F_0 \leftarrow \gamma$
2: Learn upto M gradient steps: $m = 1$
3: **repeat**
4: $F_m \leftarrow F_{m-1}$
5: $S_m^{(p)} \leftarrow$ *ComputeGradients*$\mathcal{Y}^{(p)}, \mathcal{X}, F_m$
6: Gradients $\{\Delta_i\}_{i=1}^{|\mathcal{D}|}$, $\forall$ examples $y_i \in \mathcal{Y}^{(p)}$
7: $S', \mathcal{X}', \mathcal{Y}' \leftarrow$ *SubSampleNeg*$S_m^{(p)}, \mathcal{X}, \mathcal{Y}^{(p)}$
8: $\psi_m \leftarrow$ *LearnWDAC*$S', \mathcal{X}', \mathcal{Y}'$
9: $F_m \leftarrow F_m + \psi_m$
10: **until** $m = M$
11: Return F_m

argument p is the index to the particular query variable in $\mathcal{Y}$ for which the discriminative model will be learned. For a collective classification task the function is called successively for each query variable. Since there could potentially be multiple query variables, we denote the current query variable as $\mathcal{Y}^{(p)}$.

In the m^{th} iteration of functional-gradient boosting, we compute the functional gradients for these examples using the current model F_m and the evidences of $\mathbf{y}$ as per this model (line 5). The gradients $S_m^{(p)} = \{\langle y_i, \Delta_i \rangle\}$ (where Δ_i is actual gradient value for the example y_i) are then used to learn a new weak AC ψ_m and added to the model **[lines 6,8]**. Note, however, that SUBSAMPLENEG() **[line 6]** sub-samples from the negative examples and the corresponding gradients and evidences ($\mathcal{Y}' \subset \mathcal{Y}^{(p)} = \{y : \forall y, y \in \mathcal{Y}^{(p)}, y \in \mathcal{Y}', y = 0, y \text{ consistent with sampler}\}$, $S' = \{s_i : \forall i, s_i \in S_m^{(p)}, y_i \in \mathcal{Y}'\}$ and $\mathcal{X}' = \{\mathbf{x}_i : \forall i, \mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}'\}$). Learning a new weak AC based on the current sub-sampled gradients involves a call to the procedure LEARNWDAC() as summarized in Alg. 2.

Alg. 2 outlines the procedure LEARNWDAC, that takes the gradients and the corresponding evidence and labels (for the query variable $\mathcal{Y}$) as input. The model and the supporting data structures are initialized **[lines 2-5]**. Note that $\mathbf{w}$ refers to all the parameter nodes in a DAC. fs is a min-heap data structure used later to search for the best set of features to be considered in constructing the DAC. To learn a valid conditional weak AC to fit the gradients at the current step we have to determine the best (most informative) feature set and it is done in a step-wise fashion. First, a set of regression trees (rooted at every feature/variable in $\mathcal{X}$) are learned using weighted variance as the scoring function **[lines 6-9]**. The feature set $\mathcal{F}$ is then computed by first constructing a min-heap fs with the features from the learned regression trees, using their weighted-variance scores. We search and retrieve from min-heap, the edges with weights lower than Ω times the highest weighted variance as seen from the current

Algorithm 2 LEARNWDAC: Fit Weak Conditional AC

Require: $S, \mathcal{X}, \mathcal{Y}^{(p)}$
1: The gradient set $S = \{\langle y_i, \Delta_i \rangle\}$
2: Conditional Arithmetic Circuit $C_{\mathbf{w}} \leftarrow \emptyset$
3: Initialize empty AC; $\mathbf{w}$: set of parameters
4: Initialize min-heap $fs \leftarrow \emptyset$
5: Feature set $\mathcal{F} \leftarrow \emptyset$
6: Regression Tree set $\mathbb{T} \leftarrow \emptyset$
7: **repeat**
8: $T_j \leftarrow$ Regression Tree rooted at f_j
9: Scored using Weighted-Variance
10: $\mathbb{T} \leftarrow \mathbb{T} \cup T_j$
11: **until** $\forall$ features $f_j \in \mathcal{X}$
12: $fs \leftarrow$ *Min-Heap* $f_j : \forall\, T_j \in \mathbb{T}$, Scores
13: $\mathcal{F} \leftarrow \mathcal{F} \cup \{f\}$: select best set $\{f\} \equiv \{T\} \subset \mathbb{T}$
14: min-heap search on fs w/ threshold $\Omega * max$
15: **repeat**
16: **if** $\Delta_{cll}(f_k) > \tau$ **then**
17: Update $C_{\mathbf{w}}$ with f_k
18: Update parameters $\mathbf{w}$
19: **end if**
20: **until** each f_k in $\mathcal{F}$
21: Return C_θ

feature set given by

$$threshold = \Omega \cdot \max_{T \in \mathbb{T}} Score(T)$$

[lines 10,11]. $\mathcal{F}$ is used to construct a DAC $C_{\mathbf{w}}$, by iteratively scanning each feature f_k in the feature set and including it in the AC if the change in conditional log-likelihood $\Delta_{cll}(f_k)$ is above a positive threshold τ ($\tau > 0$) **[lines 12-17]**. Whenever a feature f_k is included in the DAC, we jointly optimize the parameters $\mathbf{w}$ after updating the initial structure. We use L-BFGS to optimize the weights in our model. Note that, for constructing/updating the AC $C_{\mathbf{w}}$ **[line 14]** we utilize the *'Split AC'* approach proposed in the ACMN algorithm (Lowd & Rooshenas, 2013).

3.3. Discussion — DACBOOST closes loops by inducing wide-and-deep DACs

Before moving on to our empirical evaluation, let us discuss DACBOOST. DACBOOST differs from the currently best discriminative learning algorithm DACLEARN (Rooshenas & Lowd, 2016). While DACLearn induces tree structured ACs, ours is capable of learning DAGs. Intuitively, their approach can be viewed as breaking some loops in the true generative model. Since we boost the learning, we "overlay" several trees and hence could potentially repair some loops, that may otherwise have been broken if a single tree-structured AC was learned. Exploring the connection to tree-reweighted bounds and/or stacking learning to deeply understand the properties of our learning algorithm is an interesting future direction. Also, we are strongly motivated by the observation that complete and valid SPNs (correspondingly ACs) can be induced by a mixture of trees (Zhao et al., 2016). Thus, our work can be seen as extending tree SPN and AC learners with boosting to learn valid DACs with the observation that valid SPNs (and ACs) are additive tree models. That is, we widen the deep DACs by boosting in features and feature combinations as needed. We verify this empirically in the next section by demonstrating that our DACBOOST algorithm learns effective DACs by achieving equal or better performance in nearly all the domains as the state-of-the-art DACLearn algorithm. By virtue of learning shorter models, our approach is significantly faster than the original learning algorithm.

Data sets	# Variables	# Train Egs
ADNI	29	350
DDI	25	16000
DDoS	20	33650
PPD	66	130
PPMI	119	1680

Table 1. The number of variables including the query and evidence variables in the real data sets.

4. Empirical Evaluation

We aim to answer the following questions explicitly in our experimental evaluations:

(Q1) Is DACBOOST competitive with state-of-the-art in terms of predictive performance as measured by conditional log-likelihood?

(Q2) Can boosting DACs actually be faster than learning a single DAC?

To this end, we implemented DACBOOST as an extension to the DACLEARN code-base available as a part of the open-source Libra toolkit[1] (Lowd & Rooshenas, 2015); consequently, DACBOOST inherits all the system requirements and the library dependencies as the original DACLEARN. We evaluated DACBOOST and DACLEARN on both real and standard data sets compared to state-of-the-art discriminative structure learning algorithm for ACs, DACLearn. All experiments were conducted on *Intel(R) Xeon(R) CPU E5-2630 v3* server machines, clocking @ 2.40GHz and usable memory of 235GB.

4.1. Data Sets Description

We investigated **(Q1)**, **(Q2)** on both standard domains and some novel domains for probabilistic modeling. Specifically,

[1] http://libra.cs.uoregon.edu/

we chose four clinical/medical data sets, a network traffic for DDoS attack detection data set and some standard data sets. We now describe the data sets briefly focusing on the novel ones:

1. **Alzheimer's**: The Alzheimer's Disease NeuroIntiative (ADNI[2]) is designed to verify whether MRI and PET images, genetics, cognitive tests and blood biomarkers can be used for early prediction Alzheimers disease. We learn a DAC for modeling Alzheimer's vs cognitively normal, conditioned on the demographics features and MMScore (cognitive test score).
2. **Drug-Drug interactions (DDI)**: This data set consists of 78 drugs obtained from (DrugBank[3]). The goal is to learn a distribution of drug-drug interactions conditioned on the chemical pathways (Dhami et al., 2018).
3. **DDoS attack detection (DDoS)**: Employed in the work of Ricks et al. (Ricks et al., 2018b), benign and large-scale botnet network traffic is captured for use in DDoS attack detection. A key aspect is the automation of client-side human behavior for generation of benign network traffic in a manner scalable to network size (Ricks et al., 2018a). Our goal is to learn a CAC model for attack modeling given benign and botnet network traffic.
4. **Post-Partum Depression (PPD)**: Inspired by the work of Natarajan et al (Natarajan et al., 2017), the goal is to model post-partum depression diagnosis based on online questionnaire data including demographics, family history (relationship), social support, economic status, infant behavior and CDC questions.
5. **Parkinson's**: Parkinson's Progression Markers Initiative (PPMI[4]) is a study designed to identify biomarkers that impact Parkinson's progression in a subject. Features include imaging data, clinical data, biospecimens, demographics and Montreal Cognitive Assessment Score (MoCA) and the goal is to learn the conditional distribution of occurrence of PPMI (Dhami et al., 2017).

The number of variables and training examples in each of these novel data sets are summarized in Tab. 1. In addition, we employed the benchmark data sets that were extensively used in prior work on learning SPNs and ACs (Rahman et al., 2014; Davis & Domingos, 2010; Gens & Domingos, 2013; Rooshenas & Lowd, 2014; 2016). The goal was to have mix of both established benchmarks where DACLEARN has state-of-the-art performance and some novel domains for learning discriminative probabilistic models.

[2] www.loni.ucla.edu/ADNI
[3] https://www.drugbank.ca/
[4] www.loni.ucla.edu/PPMI

4.2. Experimental Protocol

Following the experimental protocol of the previous work (Rooshenas & Lowd, 2016), we created train and test sets (with 80-20 split). For DACLearn, we used the parameterization as suggested in the paper with L1 prior of 0.1, 0.5, 1, and 2, and feature penalties of 2, 5, and 10, and an edge penalty of 0.1, a maximum circuit size of 1M edges, and a feature batch size of 2. In some datasets, we bounded the learning and inference time of DACLearn method to 300 times that of the reported learning and inference time of DACBoost in our experiments. For DACBOOST, based on the training conditional log-likelihood, we chose the number of weak DACs to be between 3 and 8.

4.3. (Q1&2) Predictive Performance and Run-Time

The predictive performances and running times on the 5 novel data sets are summarized in Tab. 2. As can be seen, it is fairly clear that DACBOOST is at least as good as or better than DACLearn in several of these data sets (4/5). This provides an affirmative answer to **(Q1)**. More importantly, it is faster in the majority of the data sets and in some cases even an order of magnitude faster. This answers **(Q2)** affirmatively.

To understand the differences, we focus on specific data sets. Consider the lower performance in DDI data set. On inspection, this data set is quite sparse. There are several features whose values are 0 uniformly across all the examples. Our hypothesis is that such features are not specifically useful for constructing weak ACs and hence boosting does not perform as well as DACLearn (even though it is not significantly worse). In DDoS data set, many of the examples are repeated and the amount of these repetitions is high. In DACBOOST, in Alg. 2, it can be observed that we subsample the negatives. This subsampling process, with large number of repetitions, can be ineffective, thus explaining the slower convergence rate (even though it converges with minor improvement over DACLearn). Finally, in the cases where DACBOOST is better, for instance in ADNI, the data set is quite clean with nearly no missing/repeated values. Both algorithms have reasonably good performance while DACBOOST is better in terms of training time (nearly half of DACLearn). Over all, DACBOOST performs equally or better than the strong baseline in majority of the data sets.

The results on the benchmark data sets used previously in learning ACs and DACs yield a clearer picture. The first key observation across **all** data sets is that, the performance of DACBOOST in terms of learning time is significantly faster than the state-of-the-art in most cases. In some domains such as MSNBC and Plants, the learning time of DACBOOST is order-of-magnitude smaller. This allows us to answer **(Q2)** affirmatively. The boosted approach can be faster than learning a single AC by virtue of learning smaller

Data sets	DACBOOST		DACLearn		Speedup
	CLL	Time (Sec.)	CLL	Time (Sec.)	
ADNI	**-0.178**	**0.950**	-0.180	1.90	2.000
DDI	-0.264	**133.87**	**-0.245**	158.86	1.186
DDoS	**-0.018**	133.52	-0.019	**121.11**	0.907
PPD	**-0.411**	**12.19**	-0.785	13.83	1.134
PPMI	**-0.163**	**353.74**	-0.188	522.99	1.478

Table 2. Evaluation results of DACBOOST on real domains (data sets). Conditional log-likelihood, *CLL*, illustrates the effectiveness (higher ⇒ better), while the running time, *Time(Sec.)* shows the efficiency (lower ⇒ better). *Speedup* is the ratio of the running time of DACLearn to that of DACBOOST.

Data sets	DACBOOST		DACLearn		Speedup
	CLL	Time (Sec.)	CLL	Time (Sec.)	
Jester	**-0.409**	**481.06**	-0.665	9851.41	20.478
KDDCup	**-0.073**	**756.68**	**-0.073**	1248.77	1.650
Kosarek	**-0.011**	**1355.84**	-0.021	3778.92	2.787
MSNBC	**-0.108**	**76.61**	-0.110	12153.90	158.640
NLTCS	**-0.148**	**17.01**	-0.152	40.35	2.372
Plants	-0.298	**467.60**	**-0.255**	13077.74	27.967
WebKB	**-0.178**	**6154.40**	-0.293	17840.81	2.900

Table 3. Results of DACBOOST on benchmark data sets that have been used and reported in DACLEARN (Rooshenas & Lowd, 2016). Conditional log-likelihood, *CLL*, shows the effectiveness (higher ⇒ better), while the running time, *Time(Sec.)* shows the efficiency (lower ⇒ better). *Speedup* is the ratio of the running time of DACLearn to that of DACBOOST.

ACs. To answer **(Q1)**, when observing the CLL values in the table, it can be easily observed that the boosting approach is equal or better in nearly all the domains. The one domain where it is worse is Plants and in that domain, the efficiency is significantly higher (about 28 times faster) for a small loss in CLL. Across all the domains, it can be stated that the boosting approach is indeed more efficient (with speedups ranging between 1.65 to 159) and equally effective when compared to learning single ACs. Specifically, on the largest data set (MSNBC with 219k examples), the difference is significant in terms of learning time illustrating the potential of DACBOOST on large data sets.

In summary, the empirical evaluations on novel and established benchmark data sets appear to clearly demonstrate the potential for learning DACs in a stage-wise manner. Even in domains where there is a small loss in performance, the learning time is significantly faster than learning a full DAC. Construction of a full DAC from these boosted DACs is an interesting and immediate direction for future analysis. In any case, all the questions can be answered affirmatively.

5. Conclusions

Arithmetic circuits emphasize the important role of depth in learning tractable probabilistic models. In this paper, we argued that width is equally important. Unfortunately, wide and deep probabilistic models are more difficult to train. We presented the first boosting framework to ease the training of tractable discriminative probabilistic models, specifically conditional ACs. We derived the functional gradients of the examples, outlined the method for learning weak ACs and presented the algorithm, called DACBOOST, for learning them given the data. Our empirical evidence shows that boosted conditional ACs can gain predictive performance, sometimes in an fraction of time.

There are several avenues for future work: (1) analysis of the theoretical properties including bounds and convergence, (2) a more comprehensive evaluation on all the standard data sets used in (Rooshenas & Lowd, 2016). Here, carrying over ideas of residuals networks (He et al., 2016) to ACs appears to be promising. Using domain-specific human input as an inductive bias could make the algorithm converge even faster. Finally, how to make a broader class of tractable probabilistic models including generative models wider and deeper remains an open question from both a theoretical as well as an algorithmic perspective.

Acknowledgements. The authors thank the anonymous reviewers. KK acknowledges the support of the DFG project CAML (KE 1686/3-1, SPP 1999) and of the RMU project DeCoDeML. NR & SN acknowledge the support of relationalAI. MD & SN gratefully acknowledge the support of CwC Program Contract W911NF-15-1-0461 with the US Defense Advanced Research Projects Agency (DARPA) and the Army Research Office (ARO).

References

Bach, F. R. and Jordan, M. I. Thin junction trees. In *NIPS*, 2002.

Bouman, C. A. and Shapiro, M. A multiscale random field model for bayesian image segmentation. *IEEE Transactions on image processing*, 1994.

Chandrasekaran, V., Srebro, N., and Harsha, P. Complexity of inference in graphical models. In *UAI*, 2008.

Chen, T., Singh, S., Taskar, B., and Guestrin, C. Efficient second-order gradient boosting for conditional random fields. In *AISTATS*, 2015.

Darwiche, A. A differential approach to inference in bayesian networks. *Journal of the ACM*, 2003.

Davis, J. and Domingos, P. Bottom-up learning of markov network structure. In *ICML*, 2010.

Dhami, D. S., Soni, A., Page, D., and Natarajan, S. Identifying parkinsons patients: A functional gradient boosting approach. In *AIME*, 2017.

Dhami, D. S., Kunapuli, G., Das, M., Page, D., and Natarajan, S. Drug-drug interaction discovery: Kernel learning from heterogeneous similarities. *Smart Health*, 2018.

Dietterich, T., Ashenfelter, A., and Bulatov, Y. Training CRFs via gradient tree boosting. In *ICML*, 2004.

Dietterich, T., Hao, G., and Ashenfelter, A. Gradient tree boosting for training conditional random fields. *JMLR*, 2008.

Friedman, J. Greedy function approximation: A gradient boosting machine. *Annals of Statistics*, 2001.

Gens, R. and Domingos, P. Discriminative learning of sum-product networks. In *NIPS*, 2012.

Gens, R. and Domingos, P. Learning the structure of sum-product networks. In *ICML*, 2013.

Gogate, V., Webb, W., and Domingos, P. Learning efficient markov networks. In *NIPS*, 2010.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Karger, D. and Srebro, N. Learning markov networks: Maximum bounded tree-width graphs. In *SODA*, 2001.

Khot, T., Natarajan, S., Kersting, K., and Shavlik, J. Learning Markov logic networks via functional gradient boosting. In *ICDM*, 2011.

Lowd, D. and Domingos, P. Learning arithmetic circuits. In *UAI*, 2008.

Lowd, D. and Rooshenas, A. Learning markov networks with arithmetic circuits. In *AISTATS*, 2013.

Lowd, D. and Rooshenas, A. The libra toolkit for probabilistic models. *JMLR*, 2015.

Munoz, D., Vandapel, N., and Hebert, M. Directional associative markov network for 3-d point cloud classification. In *3DPVT*, 2008.

Natarajan, S., Khot, T., Kersting, K., Gutmann, B., and Shavlik, J. Gradient-based boosting for statistical relational learning: The relational dependency network case. *MLJ*, 2012.

Natarajan, S., Prabhakar, A., Ramanan, N., Baglione, A., Connelly, K., and Siek, K. Boosting for postpartum depression prediction. In *CHASE*, 2017.

Osokin, A., Vetrov, D., and Kolmogorov, V. Submodular decomposition framework for inference in associative markov networks with global constraints. In *CVPR*, 2011.

Poon, H. and Domingos, P. Sum-product networks: A new deep architecture. In *UAI*, 2011.

Rahman, T., Kothalkar, P., and Gogate, V. Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of chow-liu trees. In *ECMLPKDD*, 2014.

Ramanan, N., Kunapuli, G., Khot, T., Fatemi, B., Kazemi, S. M., Poole, D., Kersting, K., and Natarajan, S. Structure learning for relational logistic regression: An ensemble approach. In *KR*, 2018.

Ricks, B., Tague, P., and Thuraisingham, B. Large-scale realistic network data generation on a budget. In *IRI*, 2018a.

Ricks, B., Thuraisingham, B., and Tague, P. Lifting the smokescreen: Detecting underlying anomalies during a ddos attack. In *ISI*, 2018b.

Rooshenas, A. and Lowd, D. Learning sum-product networks with direct and indirect variable interactions. In *ICML*, 2014.

Rooshenas, A. and Lowd, D. Discriminative structure learning of arithmetic circuits. In *AISTATS*, 2016.

Taskar, B., Chatalbashev, V., and Koller, D. Learning associative markov networks. In *ICML*, 2004.

Vernaza, P., Taskar, B., and Lee, D. D. Online, self-supervised terrain classification via discriminatively trained submodular markov random fields. In *ICRA*, 2008.

Yang, S., Khot, T., Kersting, K., and Natarajan, S. Learning continuous-time bayesian networks in relational domains: A non-parametric approach. In *AAAI*, 2016.

Zhao, H., Poupart, P., and Gordon, G. J. A unified approach for learning the parameters of sum-product networks. In *NIPS*, 2016.