Learning Logistic Circuits

Yitao Liang and **Guy Van den Broeck**
Computer Science Department
University of California, Los Angeles
{yliang, guyvdb}@cs.ucla.edu

Abstract

This paper proposes a new classification model called logistic circuits. On MNIST and Fashion datasets, our learning algorithm outperforms neural networks that have an order of magnitude more parameters. Yet, logistic circuits have a distinct origin in symbolic AI, forming a discriminative counterpart to probabilistic-logical circuits such as ACs, SPNs, and PSDDs. We show that parameter learning for logistic circuits is convex optimization, and that a simple local search algorithm can induce strong model structures from data.

1 Introduction

Circuit representations are a promising synthesis of symbolic and statistical methods in AI. They are "deep" layered data structures with statistical parameters, yet they also capture intricate structural knowledge. Recently, many representations have been proposed for learning tractable probability distributions: arithmetic circuits (Lowd and Domingos 2008), weighted SDD (Bekker et al. 2015), PSDD (Kisa et al. 2014), cutset networks (Rahman, Kothalkar, and Gogate 2014) and sum-product networks (SPNs) (Poon and Domingos 2011). Collectively, these approaches achieve the state of the art in discrete density estimation and vastly outperform classical probabilistic graphical model learners (Gens and Domingos 2013; Rooshenas and Lowd 2014; Adel, Balduzzi, and Ghodsi 2015; Rahman and Gogate 2016; Liang, Bekker, and Van den Broeck 2017). However, we have not observed the same success when deploying circuit representations for *classification or discriminative learning*. Probabilistic circuit classifiers significantly lag behind the performance of neural networks (Benenson 2018).

In this paper, we propose a new classification model called *logistic circuits*, which shares many syntactic properties with the representations mentioned earlier. One can view logistic circuits as the discriminative counterpart to probabilistic circuits. Owing to their elegant properties, learning the parameters of a logistic circuit can be reduced to a logistic regression problem and is therefore convex. Learning logistic circuit structure is reduced to a simple local search problem using primitives from the probabilistic circuit learning literature (Liang, Bekker, and Van den Broeck 2017).

We run experiments on standard image classification benchmarks (MNIST and Fashion) and achieve accuracy higher than much larger MLPs and even CNNs with an order of magnitude more parameters. For example, logistic circuits obtain 99.4% accuracy on MNIST. Compared to other tractable learners on MNIST, and the state-of-the-art discriminative SPN learner in particular (Peharz et al. 2018), our logistic circuit learner cuts the error rate by a factor of three. Furthermore, we show our learner is highly data efficient, managing to still learn well with limited data.

This paper proceeds as follows. Section 2 introduces the syntax and semantics of logistic circuits. Sections 3 and 4 describe our parameter and structure learning algorithms, which Section 5 evaluates empirically. Section 6 elaborates on the connection with tractable generative models, after which we conclude with related and future work.

2 Representation

This section introduces the logistic circuit representation.

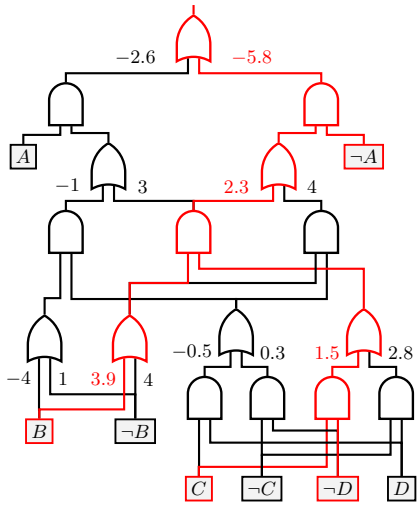
Notation We use uppercase X to denote a Boolean random variable and lowercase x for a specific assignment to it. Interchangeably, we also interpret Boolean random variables as logical variables. A set of variables $\mathbf{X}$ and their joint assignments $\mathbf{x}$ are denoted in bold. A complete assignment $\mathbf{x}$ to all variables is a possible world, or interchangeably, a data sample. Literals are variables X or their negation $\neg X$. Logical sentences are constructed from literals and connectives such as AND and OR in the standard way. An assignment $\mathbf{x}$ that satisfies a logical sentence α is denoted as $\mathbf{x} \models \alpha$.

2.1 Logical Circuits

A logical circuit is a directed acyclic graph representing a logical sentence, as depicted in Figure 1a (ignoring parameters for now). Each inner node is either an AND gate or an OR gate.[1] A leaf (input) node represents a Boolean literal, that is, X or $\neg X$, where the node can only be satisfied if X is set to 1 (true) respectively 0 (false).

The following properties are key for logical circuits to be well-behaved (Darwiche and Marquis 2002). An AND gate

[1] We consider negation-normal-form circuits where no negation is allowed except at the leafs/inputs (Darwiche and Marquis 2002).



(a) Logistic circuit

A	B	C	D	$g_r(ABCD)$	$\Pr(Y=1 \mid ABCD)$
1	0	1	1	-3.1	4.31%
0	1	1	0	1.9	86.99%
1	1	1	0	5.8	99.70%

(b) Weights and classification probabilities for select examples

Figure 1: A logistic circuit with example classifications.

is *decomposable* if its inputs depend on disjoint sets of variables. For example, the top-most AND gates in Figure 1a depend on A in their one input and on $\{B, C, D\}$ in their other input. When an AND gate has two inputs, they are called its prime (left) and sub (right). An OR gate is *deterministic* if for any single complete assignment, at most one of its inputs can be set to 1. For example, the left input to the root OR gate in Figure 1a is 1 precisely when $A = 1$, and its other input is 1 precisely when $A = 0$.

Logical circuits can be extended to *probabilistic circuits* that represents a probability distribution over binary random variables, for example by parameterizing wires with conditional distributions (Kisa et al. 2014). Probabilistic circuits have been successfully used for generative learning (Liang, Bekker, and Van den Broeck 2017). Section 6 will discuss probabilistic circuits in more detail.

2.2 Logistic Circuits

This paper proposes *logistic circuits* for classification. Syntactically, they are logical circuits where every AND is decomposable and every OR is deterministic. However, logistic circuits further associate real-valued parameters $\theta_1, \ldots, \theta_m$ with the m input wires to every OR gate. For example, the root OR node in Figure 1a associates parameters -2.6 and -5.8 with its two inputs.

To give semantics to logistic circuits, we first characterize how a particular complete assignment $\mathbf{x}$ (one data example) propagates through the circuit.

Definition 1 (Boolean Circuit Flow)**.** *Consider a deterministic OR gate n. The Boolean flow $f(n, \mathbf{x}, c)$ of a complete assignment $\mathbf{x}$ between parent n and child c is*

$$f(n, \mathbf{x}, c) = \begin{cases} 1 & \text{if } \mathbf{x} \models c \\ 0 & \text{otherwise} \end{cases}$$

For example, under the assignment $A = 0, B = 1, C = 1, D = 0$, the root node in Figure 1a has a Boolean circuit flow of 0 with its left child and 1 with its right child. Note that the determinism property guarantees that under every OR gate, for a given example $\mathbf{x}$, at most one wire has a flow of 1, and the rest has a flow of 0.

We are now ready to define the logistic circuit semantics.

Definition 2 (Logistic Circuit Semantics)**.** *A logistic circuit node n defines the following weight function $g_n(\mathbf{x})$.*

- *If n is a leaf (input) node, then $g_n(\mathbf{x}) = 0$.*
- *If n is an AND gate with children $c_1, \ldots, c_m$, then*

$$g_n(\mathbf{x}) = \sum_{i=1}^{m} g_{c_i}(\mathbf{x}).$$

- *If n is an OR gate with (child node, wire parameter) inputs $(c_1, \theta_1), \ldots, (c_m, \theta_m)$, then*

$$g_n(\mathbf{x}) = \sum_{i=1}^{m} f(n, \mathbf{x}, c_i) \cdot \left(g_{c_i}(\mathbf{x}) + \theta_i\right).$$

At root node r with weight function $g_r(\mathbf{x})$, the logistic circuit defines the posterior distribution on class variable Y as

$$\Pr(Y = 1 \mid \mathbf{x}) = \frac{1}{1 + \exp\left(-g_r(\mathbf{x})\right)}. \tag{1}$$

Using Boolean circuit flow, this definition essentially collects all the parameters on wires with flow 1 that reach the root, in order to then make a prediction. This is illustrated in Figure 1a by coloring red the gates and wires whose parameters and weight function are propagated upward for the example assignment $A = 0, B = 1, C = 1, D = 0$. The logistic circuit in Figure 1a defines the same posterior predictions as the table in Figure 1b. Specifically, for the example assignment, the weight function simply sums the parameters colored in red: $-5.8 + 2.3 + 3.9 + 1.5 = 1.9$. We then apply the logistic function (Eq. 1) to get the classification probability $\Pr(Y = 1 \mid \mathbf{x}) = \frac{1}{1+\exp(-1.9)} = 86.99\%$.

2.3 Real-Valued Data

The semantics given so far assume Boolean inputs $\mathbf{x}$, which is a rather restrictive assumption and prohibits many machine learning applications. Next, we augment the logistic circuit semantics such that they can classify examples with continuous variables.

We interpret real-valued variables $q \in [0, 1]$ as parameterizing an (independent) Bernoulli distribution (cf. Xu et al. (2018)). Each continuous variable represents the probability of the corresponding Boolean random variable X. For example, with $\mathbf{q}$ setting $A = 0.4$, $B = 0.8$, $C = 0.2$, and $D = 0.7$, the probability of $\neg A \wedge D$ would be $(1-0.4)\cdot 0.7 = 0.42$. The same distribution defines a probability for each logical sentence, and therefore each node in the logistic circuit. This allows us to generalize Boolean flow as follows.

Definition 3 (Probabilistic Circuit Flow)**.** *Consider a deterministic OR gate n. Let $\mathbf{q}$ be a vector of probabilities, one for each variable in $\mathbf{X}$. The probabilistic flow $f(n, \mathbf{q}, c)$ of vector $\mathbf{q}$ between parent n and child c is*

$$f(n, \mathbf{q}, c) = \Pr_{\mathbf{q}}(c \mid n) = \frac{\Pr_{\mathbf{q}}(c \wedge n)}{\Pr_{\mathbf{q}}(n)} = \frac{\Pr_{\mathbf{q}}(c)}{\Pr_{\mathbf{q}}(n)},$$

where $\Pr_{\mathbf{q}}(.)$ is the fully-factorized distribution where each variable in $\mathbf{X}$ has the probability assigned by $\mathbf{q}$.

Logistic circuit semantics now support continuous data (after normalizing to $[0, 1]$), simply by replacing Boolean flow with probabilistic flow in Definition 2. Note that probabilistic circuit flow has Boolean circuit flow as a special case, when $\mathbf{q}$ happens to be binary. Furthermore, due to the determinism and decomposability properties, the probabilities in Definition 3 can be computed efficiently, together with all probabilistic circuit flows and weight functions in the logistic circuit. We defer the discussion of these computational details to Section 3.4. In the rest of this paper, we will abuse notation and have $\mathbf{x}$ refer to Boolean inputs as well as continuous inputs $\mathbf{q}$ interchangeably.

3 Parameter Learning

A natural next question is how to learn logistic circuit parameters from complete data, for a fixed given circuit structure (structure learning is discussed in Section 4). Furthermore, we ask whether those learned parameters are guaranteed to be optimal, globally minimizing a loss function. We address these questions by showing how parameter learning can be reduced to logistic regression on a modified set of features, owing to logistic circuits' strong properties.

3.1 Special Cases

Before presenting the general reduction, we briefly discuss two special cases that establish some intuition.

Linear Weight Functions Consider a vanilla logistic regression model on input variables (features) $\mathbf{X}$. Does there exist an equivalent logistic circuit with the same weight function? For sample $\mathbf{x}$, logistic regression with parameters $\boldsymbol{\theta}$ would have weight function $\mathbf{x} \cdot \boldsymbol{\theta}$. Following Definition 2, we obtain such a simple weight function (linear in the input variables) by placing OR gates over complementary pairs of literals and associating a θ parameter which each wire (see Figure 2).[2] A large parent AND gate collects these variable-wise weights into a single linear sum. Finally, an OR gate at the root adds the bias term regardless of the input.

Proposition 1. *For each classical logistic regression model, there exists an equivalent logistic circuit model.*

Boolean Flow Indicators Next, let us consider a special case that makes no assumptions about circuit structure, but that requires the inputs to be fully binary. Such a circuit would have Boolean flows through every wire. Instead of working with the input variables $\mathbf{X}$, we can introduce new features that are indicator variables, telling us how the example propagates through the circuit, and which wires have a Boolean flow that reaches the circuit root. The circuit flows (indicators) decide which parameters are summed into the weight function; this process has been implicitly revealed in Figure 1a. By introducing such indicators, we can always obtain a linear weight function of composite features that are extracted from sample $\mathbf{x}$. Next, we generalize this idea of introducing wire features to arbitrary logistic circuits.

[2] The negated variable inputs and parameters $\theta_{\neg X}$ are redundant, but we keep them for the sake of consistency. Alternatively, we can fix $\theta_{\neg X} = 0$ for all X to remove this redundancy.

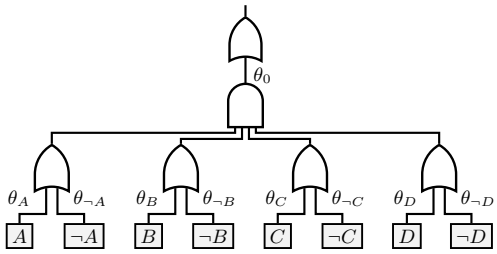


Figure 2: Logistic regression represented as a logistic circuit

3.2 Reduction to Logistic Regression

We will now consider the most general case, with continuous input data and no assumptions on the circuit structure.

Proposition 2. *Any logistic circuit model can be reduced to a logistic regression model over a particular feature set.*

Corollary 3. *Logistic circuit cross-entropy loss is convex.*

To prove Proposition 2, we need to rewrite the classification distribution in Definition 2 as follows.

$$\Pr(Y = 1 \mid \mathbf{x}) = \frac{1}{1 + \exp(-\mathbb{x} \cdot \boldsymbol{\theta})}.$$

Here, $\mathbb{x}$ is some vector of features extracted from the raw example $\mathbf{x}$. This feature vector can only depend on $\mathbf{x}$; not on the parameters $\boldsymbol{\theta}$. Thus, the fundamental question is whether we can decompose $g_n(\mathbf{x})$ into $\mathbb{x} \cdot \theta$ for all nodes n. We prove this to be true by induction:

- Base case: n is a leaf (input) node. It is obvious g_n can be expressed as $\mathbb{x} \cdot \theta$ since g_n always equals 0.
- Induction step: assume g of all the nodes under node n can be expressed as $\mathbb{x} \cdot \theta$. We need to consider two cases:
  1. If n is an AND gate having (w.l.o.g.) two children, prime p and sub s. Given $g_p = \mathbb{x}_p \cdot \theta_p$ and $g_s = \mathbb{x}_s \cdot \theta_s$,

$$\begin{aligned} g_n &= \mathbb{x}_p \cdot \theta_p + \mathbb{x}_s \cdot \theta_s \\ &= \begin{bmatrix} \mathbb{x}_p \\ \mathbb{x}_s \end{bmatrix} \cdot \begin{bmatrix} \theta_p \\ \theta_s \end{bmatrix}. \end{aligned}$$

  2. If n is an OR gate with (child node, wire parameter)

inputs $\{(c_1, \theta_1), \ldots, (c_m, \theta_m)\}$. Given $g_{c_i} = \mathbb{x}_{c_i} \cdot \theta_{c_i}$,

$$g_n = \sum_i f(n, \mathbf{x}, c_i) \cdot (\mathbb{x}_{c_i} \cdot \theta_{c_i} + \theta_i)$$

$$= \begin{bmatrix} f(n, \mathbf{x}, c_1) \cdot \mathbb{x}_{c_1} \\ f(n, \mathbf{x}, c_1) \\ \vdots \\ f(n, \mathbf{x}, c_m) \cdot \mathbb{x}_{c_m} \\ f(n, \mathbf{x}, c_m) \end{bmatrix} \cdot \begin{bmatrix} \theta_{c_1} \\ \theta_1 \\ \vdots \\ \theta_{c_m} \\ \theta_m \end{bmatrix}.$$

Note that this proof holds true regardless of whether the input sample $\mathbf{x}$ is binary or real-valued. With this proof, it is obvious that learning the parameters of a logistic circuit is equivalent to logistic regression on features $\mathbb{x}$. We refer readers to Rennie (2005) for a detailed proof that logistic regression is convex.

Given this correspondence, any convex optimization technique can now be brought to bear on the problem of learning the parameters of a logistic circuit. In particular, we use stochastic gradient descent for this task.

3.3 Global Circuit Flow Features

In the proof of Proposition 2, features $\mathbb{x}$ are computed recursively by induction. However, it is not clear what these features represent, and how they are connected to the input data. In this section we assign semantics to those extracted features. They are the *global circuit flow* of the observed example through the circuit. Global circuit flow is defined with respect to the root of a logistic circuit.

Definition 4 (Global Circuit Flow)**.** *Consider a logistic circuit over variables $\mathbf{X}$ rooted at OR gate r. The global circuit flow $f_r(n, \mathbf{x}, c)$ of input $\mathbf{x}$ between parent n and child c is defined inductively as follows. The global circuit flow between root r and its child c is the (local) probabilistic circuit flow: $f_r(r, \mathbf{x}, c) = f(r, \mathbf{x}, c)$. Then, for any node n with parents $v_1, \ldots, v_m$, we have that*

– *if n is an AND gate, global flow from child c is*

$$f_r(n, \mathbf{x}, c) = \sum_{i=1}^{m} f_r(v_i, \mathbf{x}, n),$$

– *if n is an OR gate, global flow from child c is*

$$f_r(n, \mathbf{x}, c) = f(n, \mathbf{x}, c) \cdot \sum_{i=1}^{m} f_r(v_i, \mathbf{x}, n).$$

The red wires in Figure 1a have a global circuit flow of 1 for the given Boolean input. In general, global circuit flow assigns a continuous probability value to each wire.

Based on global circuit flow, we postulate the following alternative semantics for logistic circuits.

Definition 5 (Logistic Circuit Alternative Semantics)**.** *Let $\mathcal{W}$ be the set of all wires (n, θ, c) between OR gates n and children c with parameters θ. Then, a logistic circuit rooted at node r defines the weight function*

$$g_r(\mathbf{x}) = \sum_{(n,\theta,c) \in \mathcal{W}} f_r(n, \mathbf{x}, c) \cdot \theta.$$

Note that the definition of global circuit flows, as well as our alternative semantics, follow a top-down induction. In contrasts, the original semantics in Definition 2 follow a bottom-up induction. We resolve this discrepancy next.

Proposition 4. *The features $\mathbb{x}$ constructed in the proof of Proposition 2 are equivalent to global flows $f_r(n, \mathbf{x}, c)$.*

Corollary 5. *The bottom-up semantics of Definition 2 and the top-down semantics of Definition 5 are equivalent.*

We defer the proof of this proposition to Appendix A.

Recall that without parameters, a logistic circuit is simply a logical circuit, which means that gates in a logistic circuit have real meaning: they correspond to some logical sentence. Hence, the values of global circuit flow features $\mathbb{x}$ correspond to probabilities of these logical sentences according to the input vector $\mathbf{x}$. This provides us with a precious opportunity to assign meaning to the features learned by logistic circuits. We will revisit this point in Section 5.4, where we also visualize some global circuit flow features.

3.4 Computing Global Flow Features Efficiently

While logistic circuit parameter learning is convex, we would like to also guarantee that the required feature computation is tractable. This section discusses efficient methods to calculate global flow features $\mathbb{x}$ (i.e., $f_r(n, \mathbf{x}, c)$) from training samples $\mathbf{x}$ offline, before parameter learning.

As is clear from Definition 3, circuit flows make extensive use of node probabilities. We design our computation to consist of two parts, and dedicate the first part to the calculation of node probabilities. The first part is a bottom-up linear pass over the circuit starting with leaf nodes whose probabilities are directly provided by the input sample; see the details in Appendix B. The second part makes use of these node probabilities to calculate the global circuit flow features in linear time. It is a top-down implementation of the recursion in Definition 4; see its details in Appendix C. Note that these computations correspond to the partial derivative computations used in arithmetic circuits for the purpose of probabilistic inference (Darwiche 2003).

Our algorithm is completely compatible with fast vector arithmetic: instead of inputting one single sample each time, one can directly supply the algorithms with a vector of samples (e.g., a mini batch), and this yields significant speedups.

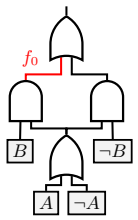
4 Structure Learning

This section presents an algorithm to learn a compact logical circuit structure for logistic circuits from data. For simplicity of designing the primitive operations, we assume AND gates always have two inputs (prime and sub).

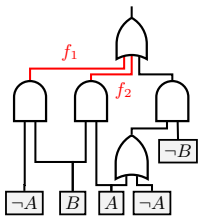
4.1 Learning Primitive

The split operation was first introduced to modify the structure of PSDD circuits (Liang, Bekker, and Van den Broeck 2017). We adopt it here with minor changes[3] as the primitive operation for our structure learning algorithm. Split-

[3] Compared to the splits in LearnPSDD (Liang, Bekker, and Van den Broeck 2017), we do not limit constraints to be on primes.



(a) Before split of f_0 on A



(b) After split of f_0 on A

A	B	f_0	f_1	f_2
1	1	1	0	1
0	1	1	1	0
0.5	0.6	0.6	0.30	0.30
0.4	0.8	0.8	0.48	0.32

(c) Circuit flow before and after the split.

Figure 3: A split changes the circuit flow.

ting an AND gate happens by imposing two additional constraints that are *mutually exclusive* and *exhaustive*, in particular by making two opposing variable assignments. Executing a split creates partial copies of the gate and some of its decedents. Furthermore, one can choose to duplicate additional nodes up to a fixed depth (3 in our experiments). We refer readers to Liang, Bekker, and Van den Broeck (2017) for further details on the algorithm for executing splits.

Splits are ideal primitives to change the classifier induced by a logistic circuit: they directly affect the circuit flows (see Figure 3). By imposing constraints on AND gates, splits alter the node probabilities associated with the affected AND gates. Following Definition 3, the circuit flows on the wires out of those AND gates adapt accordingly. While Figure 3 focuses on the immediately affected wires, the effect of a split on circuit flows can propagate downward for several levels, depending on the depth of node duplication. Still the effects of a split on both the structure of a logistic circuit and the circuit flows are very local and contained in the subcircuit rooted at the OR parent of the split AND gate. However, its effect on the parameters is global. Once a split is executed, the whole parameter set needs to be re-trained.

4.2 Learning Algorithm

The overall structure learning algorithm for logistic circuits, built on top of the split operation, proceeds as follows. Iteratively, one split is executed to change the structure, followed by parameter learning. We only consider single-variable split constraints and first select which AND gate to split, followed by a selection of which variable to split on.

When using gradient descent, one hopes that the parameter on the AND gate output consistently has its partial derivatives pointing in the same direction for all training examples. This will steadily push the parameter to a large magnitude. If this is not the case, we will use splits to alter the flow of examples through the circuit. Specifically, those AND gates whose associated output parameter has a large variance of its partial derivative (that is, the derivative of the loss function w.r.t. that parameter) requires splitting for the parameters to improve. We simply select the AND gate whose output parameter has the highest training variance.

Given an AND gate to split, we consider candidate variables X to execute the split with. We construct two sets of training examples that affect this node: in one group, each example is weighted by the marginal probability of X; in the other, with the marginal probability of $\neg X$. Next, we calculate the within-group weighted variances of the partial derivatives. The variable with the smallest weighted variances gets picked, as this suggests the split will introduce new parameters with gradients that align in one direction.

Table 1: Classification accuracy of logistic circuits in context with commonly used existing models. We report the details of those existing models in Appendix E.

Accuracy % on Dataset	Mnist	Fashion
Baseline: Logistic Regression	85.3	79.3
Baseline: Kernel Logistic Regression	97.7	88.3
Random Forest	97.3	81.6
3-layer MLP	97.5	84.8
RAT-SPN (Peharz et al. 2018)	98.1	89.5
SVM with RBF Kernel	98.5	87.8
5-Layer MLP	99.3	89.8
Logistic Circuit (binary)	97.4	87.6
Logistic Circuit (real-valued)	99.4	91.3
CNN with 3 conv layers	99.1	90.7
Resnet (He et al. 2016)	99.5	93.6

Table 2: Number of parameters of logistic circuits in context with existing SGD-based models, when achieving the classification accuracy reported in Table 1

Number of Parameters	Mnist	Fashion
Baseline: Logistic Regression	<1K	<1K
Baseline: Kernel Logistic Regression	1,521 K	3,930K
Logistic Circuit (real-valued)	182K	467K
Logistic Circuit (binary)	268K	614K
3-layer MLP	1,411K	1,411K
RAT-SPN (Peharz et al. 2018)	8,500K	650K
CNN with 3 conv layers	2,196K	2,196K
5-Layer MLP	2,411K	2,411K
Resnet (He et al. 2016)	4,838K	4,838K

5 Empirical Evaluation

In this section, we empirically evaluate the competitiveness of our learner on three aspects: classification accuracy, model complexity, and data efficiency.[4] Moreover, we visualize the most important active feature with regards to the given sample to provide local interpretation for why the learned logistic circuit makes such classification.

[4]Open-source code and experiments are available at `https://github.com/UCLA-StarAI/LogisticCircuit`.

Table 3: Comparison of logistic circuits with MLPs when trained with different percentages of the dataset.

ACCURACY % WITH % OF TRAINING DATA	MNIST			FASHION		
	100%	10%	2%	100%	10%	2%
5-LAYER MLP	99.3	**98.2**	94.3	89.8	86.5	80.9
CNN WITH 3 CONV LAYERS	99.1	98.1	95.3	90.7	87.6	83.8
LOGISTIC CIRCUIT (BINARY)	97.4	96.9	94.1	87.6	86.7	83.2
LOGISTIC CIRCUIT (REAL-VALUED)	**99.4**	97.8	**96.1**	**91.3**	**87.8**	**86.0**

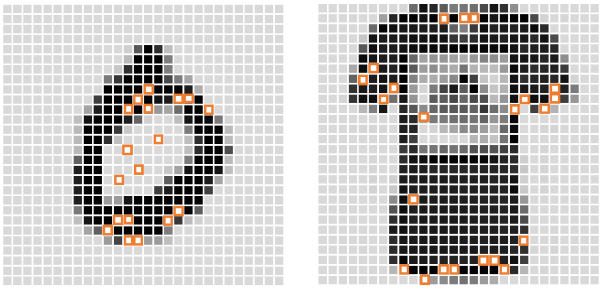


Figure 4: Visualization of the single compositional feature that contributes most to the classification probability with regards to the input image. Features are marked in orange. Left: a digit 0 from MNIST. Right: a t-shirt from Fashion.

5.1 Setup & Data Preprocessing

We choose MNIST and Fashion[5] as our testbeds. Since logistic circuits are intended for binary classification, we use the standard "one vs. rest" approach to construct an ensemble multi-class classifier such that our method can be evaluated on these two datasets. When running the binary logistic circuit, we transform pixels that are smaller than their mean plus 0.05 standard deviation to 0 and the rest to 1. When running the real-valued version, we transform pixels to $[0, 1]$ by dividing them by 255. All experiments start with a predefined initial structure; we defer its details to Appendix D. The learned structure with the highest F1 score on validation after 48 hours of running is used for evaluation. All experiments are run on single CPUs.

5.2 Classification Accuracy

Table 1 summarizes the classification accuracy on test data. Learning a logistic circuit on the binary data is on par with a 3-layer MLP; the real-valued version outperforms 5-layer MLPs and even CNNs with 3 convolutional layers. The fact that logistic circuits achieve better accuracy than CNNs is surprising, since logistic circuits do not use convolutions, which are specifically designed to exploit image invariances.

In addition, we would like to emphasis our comparison with two of the baselines. As parameter learning of logistic circuits is equivalent to logistic regression, one can view structure learning of logistic circuits as a process of constructing composite features from raw samples. The significant improvement over standard logistic regression demonstrates the effectiveness of our method in extracting valuable features; using kernel logistic regression can only partially bridge the gap in performance, yet as shown later, it does so at the cost of introducing many more parameters.

We also want to call attention to our comparison with RAT-SPN, the current state of the art in discriminative learning for probabilistic circuits. SPN is another form of circuit representation, with less restrictive structure. Parameter learning in SPN is not convex and generally requires other techniques such as EM or non-convex optimization. The empirical observation that our method achieves significantly better classification accuracy than RAT-SPN demonstrates that in structure learning, imposing more restrictions on the model's structural syntax may be beneficial. The syntactic restriction of logistic circuits requires decomposability and determinism; without them, convex parameter learning does not appear to be possible. As structure learning is built on top of parameter learning, a well-behaved parameter learning loss with a unique optimum can provide more informative guidance about how to adapt the structure, leading to a more competitive structure learning algorithm overall.

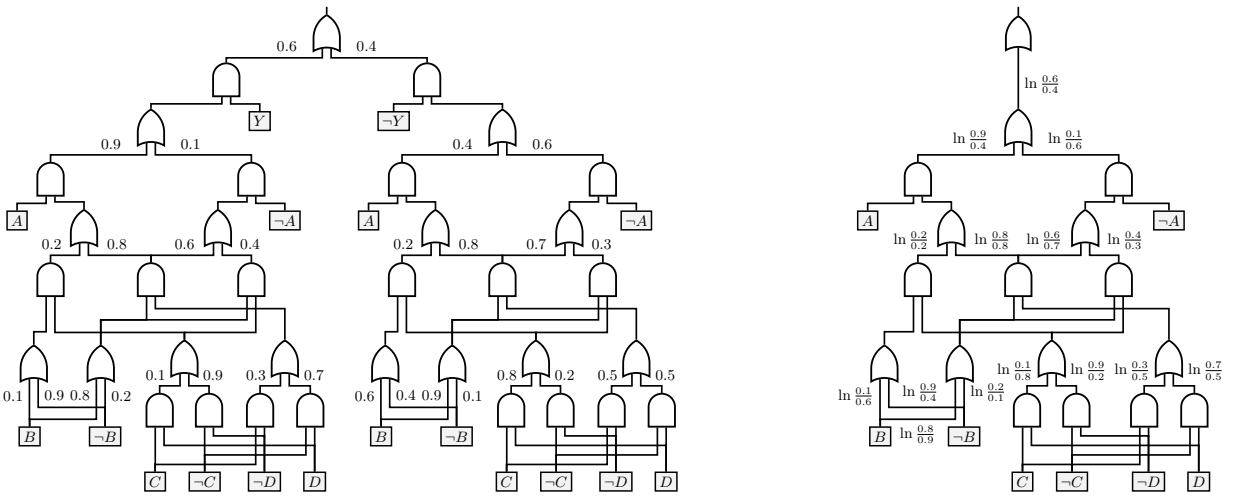
5.3 Model Complexity & Data Efficiency

Table 2 summarizes the size of all compared models when achieving the reported accuracy. We can conclude that logistic circuits are significantly smaller than the alternatives, despite attaining higher accuracy.

We design the next set of experiments to specifically investigate how well our structure learning algorithm performs under the setting where the number of training samples is limited. We have two additional sets of experiments, where only 2% and 10% of the original training data is supplied. Table 3 summarizes the performance in this limited-data setting. We mainly compare against a 5-layer MLP and CNN with 3 convolutional layers, whose performance is on par with our method under the full-data setting. As summarized in Table 3, except on MNIST with 10% training samples, real-valued logistic circuits achieve the best classification accuracy. Moreover, in both versions of logistic circuits, when the available training samples are reduced from 100% to 2%, the accuracy only drops by around 3% when evaluating on MNIST; around 5% on Fashion. In contrast, a much larger drop occurs for 5-layer MLP and CNN. Specifically, MLP's accuracy drops by 5% (9%) while CNN's accuracy drops by 4% (7%) on MNIST (Fashion). This small magnitude of accuracy decrease illustrates how data efficient our proposed structure learning algorithm is.

Except on MNIST with 10% training samples, real-valued

[5] A dataset of Zalando's images, intended as a more challenging drop-in replacement of MNIST (Xiao, Rasul, and Vollgraf 2017).



(a) Probabilistic circuit for joint distribution $\Pr(Y, A, B, C, D)$ (b) Logistic circuit for $\Pr(Y = 1 \mid A, B, C, D)$

Figure 5: A probabilistic circuit with parallel structures under class variable Y and its equivalent logistic circuit for predicting Y

logistic circuits achieve the best classification accuracy. From a top-down perspective, each OR gate of a logistic circuit presents a weighted choice between its wires. Hence, one can view a logistic circuit as a decision diagram. Under this perspective, splits refine OR gates' branching rules. As each branching rule naturally applies to multiple samples, we hypothesize that the splits selected by our structure learning algorithm reflect the general conditional feature information present in the dataset.

5.4 Local Explanation

Next, we aim to share some insights about how to explain the learned logistic circuit. Specifically, we investigate the question: "Why does the logistic circuit classify a given sample $\mathbf{x}$ as y?" Since any logistic circuit can be reduced to a logistic regression classifier, we can easily find the active global flow feature that contributes most to the given sample's classification probability. That is, the feature that maximizes $\mathbb{x} \cdot \theta$. We visualize one such feature for MNIST data and one for Fashion in Figure 4 by marking the variables used in the their corresponding logical sentences.

6 Connection to Probabilistic Circuits

In recent years, a large number of tractable probabilistic models have been proposed as a target representation for generative learning of a joint probability distribution: arithmetic circuits (Lowd and Domingos 2008), weighted SDD (Bekker et al. 2015), PSDD (Kisa et al. 2014), cutset networks (Rahman, Kothalkar, and Gogate 2014) and sum-product networks (SPNs) (Poon and Domingos 2011). These representations have various syntactic properties. Some put probabilities on terminals, others on edges. Some use logical notation (AND, OR), others use arithmetic notation ($\times$,$+$). Nevertheless, they are all circuit languages built around the properties of decomposability and/or determinism.

For our purpose, we consider a simple probabilistic circuit language based on the logistic circuit syntax, where now the θ parameters are assumed to be normalized probabilities.[6]

Definition 6 (Probabilistic Circuit Semantics)**.** *A probabilistic circuit node n defines the following joint distribution.*

- *If n is a leaf (input) node, then* $\Pr_n(\mathbf{x}) = [\mathbf{x} \models n]$.
- *If n is an AND gate with children $c_1, \ldots, c_m$, then*

$$\Pr{}_n(\mathbf{x}) = \prod_{i=1}^{m} \Pr{}_{c_i}(\mathbf{x}).$$

- *If n is an OR gate with (child node, wire parameter) inputs $(c_1, \theta_1), \ldots, (c_m, \theta_m)$, then*

$$\Pr{}_n(\mathbf{x}) = \sum_{i=1}^{m} \Pr{}_{c_i}(\mathbf{x}) \cdot \theta_i.$$

Figure 5a shows a probabilistic circuit for the joint distribution $\Pr(Y, A, B, C, D)$. This tractable circuit language is a relaxation of PSDDs (Kisa et al. 2014) and a specific type of SPN (Poon and Domingos 2011) where determinism holds throughout. It is also a type of arithmetic circuit.

We are now ready to connect logistic and probabilistic circuits. It is well known that logistic regression is the discriminative counterpart of a naive Bayes generative model (Ng and Jordan 2002). A similar correspondence holds between our logistic and probabilistic circuits.

Proposition 6. *Consider a probabilistic circuit whose structure is of the form $(Y \wedge \alpha) \vee (\neg Y \wedge \beta)$, where sub-circuits α and β are structurally identical. Then, there exists an equivalent logistic circuit for the conditional probability of Y in the probabilistic circuit. Moreover, this logistic circuit has structure $\vee\alpha$ and its parameters can be computed in closed form as log-ratios of probabilistic circuit probabilities.*

[6] We also assume *smoothness* (Darwiche and Marquis 2002).

We first depict this correspondence intuitively in Figure 5. The logistic circuit has the same structure as the two halves of the probabilistic circuit, and its parameters are computed from the probabilistic circuit probabilities. The distributions $\Pr(Y = 1 \mid A, B, C, D)$ represented by the circuits in Figures 5a and 5b are identical.

Formal Correspondence Next, we present the formal proof of this correspondence for binary $\mathbf{x}$. Recall that in our circuits, only the input wires of OR gates are parameterized. Let $\mathcal{W}_\delta$ be the set that contains all these wires in circuit δ:

$$\mathcal{W}_\delta = \{(n, c) \mid c \text{ is a gate with parent OR gate } n\}.$$

After expanding the equations in Definition 6 and following the top-down definition of global circuit flow (i.e., following Definition 4), one finds that the joint distribution induced by a probabilistic circuit δ can be rewritten as

$$\Pr_\delta(\mathbf{x}) = \prod_{(n,c)\in\mathcal{W}_\delta} f_\delta(n, \mathbf{x}, c) \cdot \theta^\delta_{(n,c)}.$$

We will exploit this finding in the derivation of the conditional distribution induced by the probabilistic circuit $\gamma = (Y \wedge \alpha) \vee (\neg Y \wedge \beta)$.

$$\begin{aligned}
&\Pr_\gamma(Y = 1 \mid \mathbf{x}) \\
&= \frac{\Pr_\gamma(Y{=}1)\Pr_\alpha(\mathbf{x})}{\Pr_\gamma(Y{=}0)\Pr_\beta(\mathbf{x}) + \Pr(Y{=}1)\Pr_\alpha(\mathbf{x})} \\
&= \frac{1}{1 + \frac{\Pr_\gamma(Y=0)\Pr_\beta(\mathbf{x})}{\Pr_\gamma(Y=1)\Pr_\alpha(\mathbf{x})}} \\
&= \frac{1}{1 + \frac{\Pr_\gamma(Y=0)\prod_{(n,c)\in\mathcal{W}_\beta} f_\beta(n,\mathbf{x},c)\theta^\beta_{(n,c)}}{\Pr_\gamma(Y=1)\prod_{(n,c)\in\mathcal{W}_\alpha} f_\alpha(n,\mathbf{x},c)\theta^\alpha_{(n,c)}}}
\end{aligned}$$

As stated in Proposition 6 and shown in Figure 5, subcircuits α and β share the same structure. Therefore, we can further simplify this equation as follows.

$$\begin{aligned}
&\Pr_\gamma(Y = 1 \mid \mathbf{x}) \\
&= \frac{1}{1 + \frac{\Pr_\gamma(Y=0)}{\Pr_\gamma(Y=1)} \prod_{(n,c)\in\mathcal{W}_\alpha} f_{\vee\alpha}(n, \mathbf{x}, c) \frac{\theta^\beta_{(n,c)}}{\theta^\alpha_{(n,c)}}} \\
&= \frac{1}{1 + \exp\left[-g(\mathbf{x})\right)]} = \Pr_{\vee\alpha}(Y = 1 \mid \mathbf{x})
\end{aligned}$$

where

$$g(\mathbf{x}) = \log \frac{\Pr_\gamma(Y{=}1)}{\Pr_\gamma(Y{=}0)} + \sum_{(n,c)\in\mathcal{W}_\alpha} f_{\vee\alpha}(n, \mathbf{x}, c) \log \frac{\theta^\alpha_{(n,c)}}{\theta^\beta_{(n,c)}} \tag{2}$$

$$= \theta^{\vee\alpha}_{root} + \sum_{(n,c)\in\mathcal{W}_\alpha} f_{\vee\alpha}(n, \mathbf{x}, c) \cdot \theta^{\vee\alpha}_{(n,c)}. \tag{3}$$

The transformation from Equation 2 to 3 expresses the logistic circuit parameters as the log-ratios of probabilistic circuit probabilities. For example, the class priors captured in the output wires of α and β are now combined as a log-ratio to form the bias term for $\vee\alpha$, expressed by the root parameter.

This proof also provides us with a new perspective to understand the semantics of the learned parameters. The parameters represent the log-odds ratio of the features given different classes. Note that by Bayes' theorem, a naive Bayes model would derive its induced distribution in a sequence of steps similar to the ones above, resulting in Equation 2. Given this correspondence, one can also view our proposed structure learning method as a way to construct meaningful features for a naive Bayes classifier. We know that after training, naive Bayes classifiers are equivalent to logistic regression classifiers (as in Equation 3).

7 Related Work

Gens and Domingos (2012) proposed the first parameter learning algorithm for discriminative SPNs, using MPE inference as a sub-routine. Without the support of the determinism property, parameter learning of general SPNs is a relatively harder question than its logistic circuit counterpart, since it is non-convex. Adel, Balduzzi, and Ghodsi (2015) boost the accuracy of SPNs on MNIST to 97.6% by extracting more representative features from raw inputs based on the Hilbert-Schmidt independence measure. Peharz et al. (2018) further improved the classification ability of SPNs by drastically simplifying SPN structure requirements and utilizing a loss objective that hybrids cross-entropy (discriminative learning) with log-likelihood (generative learning).

Rooshenas and Lowd (2016) developed a discriminative structure learning algorithm for arithmetic circuits. The method updates the circuit that represents a corresponding conditional random field (CRF) model by adding features conditioned on arbitrary evidence to the model. This work further relaxes decomposability and smoothness properties of ACs for a more compact representation. However, it targets the setting where there are a large number of output variables, not single-variable classification.

All the aforementioned literature conforms to a common trend of abandoning properties of the chosen circuit representations for easier structure learning and better prediction accuracy. They argue that those special syntactic restrictions complicate the learning process. On the contrary, this paper chooses perhaps the most structure-restrictive circuit as the target representation. Instead of relaxing the target representation's syntactical requirements, our proposed method fully leverages the valuable properties that stem from these restrictions, and in particular convexity.

8 Conclusions

We have presented logistic circuits, a novel circuit-based classification model with convex parameter learning and a simple structure learning procedure based on local search. Logistic circuits outperform much larger classifiers and perform well in a limited data regime. Compared to other symbolic, circuit-based approaches, logistic circuits present a leap in performance on image classification benchmarks. Future work includes support for convolution, parameter tying, and structure sharing in the logistic circuits framework.

Algorithm 1: Node probabilities from a real-valued sample $\mathbf{x}$.

Input: A vector of probabilities $\mathbf{x}$.
Result: $\Pr_{\mathbf{x}}(n)$: the node probability of n for $\mathbf{x}$.

1 **for** *n in the circuit's nodes, children before parents* **do**
2 **if** *n is a leaf with variable X* **then**
3 **if** *n is X* **then**
4 $\Pr_{\mathbf{x}}(n) = \mathbf{x}(X)$
5 **else**
6 $\Pr_{\mathbf{x}}(n) = 1 - \mathbf{x}(X)$
7 **else if** *n is an AND gate* **then**
8 $\Pr_{\mathbf{x}}(n) := 1$
9 **for** *c in inputs of n* **do**
10 $\Pr_{\mathbf{x}}(n) *= \Pr_{\mathbf{x}}(c)$
11 **else**
  `// n is an OR gate`
12 $\Pr_{\mathbf{x}}(n) := 0$
13 **for** *c in inputs of n* **do**
14 $\Pr_{\mathbf{x}}(n) += \Pr_{\mathbf{x}}(c)$

Algorithm 2: Features $\mathbb{x}$ from a real-valued sample $\mathbf{x}$.

Input: Node probabilities $\Pr_{\mathbf{x}}(\cdot)$.
Result: Real-valued feature vector $\mathbb{x}$.

1 **for** *n in all nodes, parents before children* **do**
2 $v(n) := 0$
3 $v(\text{root}) := 1$
4 **for** *n in all non-leaf nodes, parents before children* **do**
5 **if** *n is an OR gate* **then**
6 **for** *c in inputs of n* **do**
7 $\mathbb{x}(n, c) := v(n) \cdot \Pr_{\mathbf{x}}(c) \;/\; \Pr_{\mathbf{x}}(n)$
8 $v(c) += \mathbb{x}(n, c)$
9 **else**
  `// n is an AND gate`
10 **for** *c in inputs of n* **do**
11 $v(c) += v(n)$

A Proof of Proposition 4

Before presenting the proof, we restate the proposition.

Proposition. *The features $\mathbb{x}$ constructed in the proof of Proposition 2 are equivalent to global flows $f_r(n, \mathbf{x}, c)$.*

In the following, we prove this proposition by induction.

– Base case: the inputs of the root r are either leaf nodes or AND gates whose inputs are leaf nodes. By definition, for the root's input wires, their local circuit flow equals their global circuit flow. According to the decomposition matrix of g_n in the proof of Proposition 2, the features associated with the root's input wires are equivalent to their local circuit flow. By transitivity, we prove logistic circuits' features are equivalent to its global circuit flow vector in the base case.

– Induction step: assume the proposition holds for all OR gates in a given logistic circuit except the root r. Again, the root's inputs can be either leaf nodes or AND gates. It is obvious that for the root's input wires, their associated features are equivalent to their global circuit flow, as this has been proven in the base case. So we only need to focus on the wires of the sub logistic circuits rooted on those AND gates. The inputs to those AND gates can either be leaf nodes or OR gates. As the wires between AND gates and their leaf children do not have parameters, the correctness of the proposition does not get affected by them. We can narrow our focus again. Now let us consider an OR gate n, which is an input to some of those aforementioned AND gates $\{e_1, \ldots, e_m\}$. By our induction assumption, its features are equivalent to the global circuit flows defined with respect to n; in other words, $\mathbb{x}_n = f_n$. After propagating $\mathbb{x}_n$ upwards to the root, we get $\sum_{i=1}^{m} f(r, \mathbf{x}, e_1) \cdot \mathbb{x}_n$. The sum of the global flow on all output wires of n is $F_r(n) = \sum_{i=1}^{m} f(r, \mathbf{x}, e_1)$. Since $F_r(n)$ is propagated throughout the whole sub logistic circuit rooted at n, the global circuit flow in this sub logistic circuit with respect to the root r is $F_r(n) \cdot f_n = \sum_{i=1}^{m} f(r, \mathbf{x}, e_1) \cdot f_n$. Therefore, the constructed features are equivalent to the global circuit flows.

B Calculation of Node Probabilities

We calculate node probabilities in a bottom-up induction on the structure of the sentence.

– Base case: n is a leaf (input) node. The node probability is directly defined in $\mathbf{x}$: $\Pr_{\mathbf{x}}(n) = \mathbf{x}(X)$ if n is X; $\Pr_{\mathbf{x}}(n) = 1 - \mathbf{x}(X)$ if n is $\neg X$ (lines 2-6 in Algorithm 1).

– Induction step: given that the node probabilities for all the leaves have been calculated, we move upward to intermediate nodes and the root, where there are two cases.

  * n is an AND gate with inputs $\{c_1, \ldots, c_m\}$. Since in a logistic circuit every AND gate is decomposable, by independence of the conjuncts, $\Pr_{\mathbf{x}}(n) = \prod_{i=1}^{m} \Pr_{\mathbf{x}}(c_i)$ (lines 7-10 in Algorithm 1).
  * n is an OR gate with input nodes $\{c_1, \ldots, c_m\}$. Since every OR gate is deterministic, the probabilistic events defined at each child within the same OR parent do not intersect with each other. By mutual exclusivity, $\Pr_{\mathbf{x}}(n) = \sum_i \Pr_{\mathbf{x}}(c_i)$ (lines 11-14 in Algorithm 1).

C Calculation of Global Flows (Features)

Node probabilities $\Pr_{\mathbf{x}}(\cdot)$ are used in Algorithm 2 to obtain the final feature vector.

We perform a top-down pass starting from the root OR gate. After visiting an OR gate, the method first calculates its associated global circuit flows from its inputs; see Line 7 in Algorithm 2. These newly calculated global flows then get passed down and are accumulated on those child gates for later use on the descendent gates (Line 8). After visiting

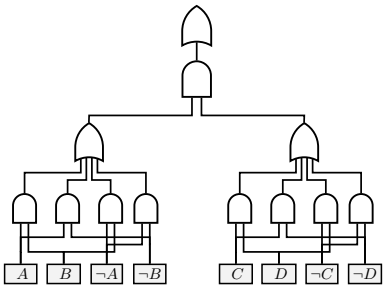


Figure 6: Initial structure of logistic circuits with 4 pixels.

an AND gate, there is no new global circuit flow to be calculated. Hence, the algorithm directly accumulates the flows passed to those AND gates to their children (Line 10-11).

Note that instead of inputing one single sample at a time, one can directly supply Algorithm 1 and 2 with a vector of samples. Our proposed calculation method is completely compatible with matrix operations, and by doing so, one can expect a large speedup.

D Initial Structure

All experiments in this paper start with an initial structure where every pixel has two corresponding leaf nodes, one for the pixel being true and the other false. Pixels are paired up by AND gates; an AND gate is created for every joint assignment to the pair. AND gates for the same pair share one OR gate parent. After this, OR gates are paired with AND gates and every AND gate is connected to its own OR gate parent until we reach the root. Figure 6 is an example of the initial structure with 4 pixels. Note that our structure learning algorithm is compatible with other initial structures and one can create ad-hoc ones tailored to different applications.

E Details of Existing Classification Models

The reported kernel logistic regression is based on the pixel n-grams implemented in Vowpal Wabbit (Langford, Li, and Strehl 2007). The reported random forest has 500 decision trees. The reported SVM with RBF Kernel uses hyper-parameters $C = 8, \gamma = 0.05$ on MNIST and $C = 4, \gamma = 25$ on Fashion. The reported 3-layer MLP has layers of size 784-1000-500-250-10 respectively. The reported 5-layer MLP has layers of size 784-1000-500-250-2000-250-10 respectively. The reported CNN with 3 convolutional layers uses 3-by-3 padded filters in the convolutional layers.

Acknowledgements

This work is partially supported by a gift from Intel, NSF grants #IIS-1657613, #IIS-1633857, #CCF-1837129, and DARPA XAI grant #N66001-17-2-4032.

References

Adel, T.; Balduzzi, D.; and Ghodsi, A. 2015. Learning the structure of sum-product networks via an svd-based algorithm. In *UAI*, 32–41.

Bekker, J.; Davis, J.; Choi, A.; Darwiche, A.; and Van den Broeck, G. 2015. Tractable learning for complex probability queries. In *NIPS*.

Benenson, R. 2018. What is the class of this image? `http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html`.

Darwiche, A., and Marquis, P. 2002. A knowledge compilation map. *JAIR* 17:229–264.

Darwiche, A. 2003. A differential approach to inference in bayesian networks. *J. ACM* 50(3):280–305.

Gens, R., and Domingos, P. 2012. Discriminative learning of sum-product networks. In *NIPS*, 3239–3247.

Gens, R., and Domingos, P. 2013. Learning the structure of sum-product networks. In *ICML*, 873–880.

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

Kisa, D.; Van den Broeck, G.; Choi, A.; and Darwiche, A. 2014. Probabilistic sentential decision diagrams. In *KR*.

Langford, J.; Li, L.; and Strehl, A. 2007. Vowpal wabbit open source project. Technical report, Yahoo!

Liang, Y.; Bekker, J.; and Van den Broeck, G. 2017. Learning the structure of probabilistic sentential decision diagrams. In *UAI*.

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

Ng, A. Y., and Jordan, M. I. 2002. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *NIPS*, 841–848.

Peharz, R.; Vergari, A.; Stelzner, K.; Molina, A.; Trapp, M.; Kersting, K.; and Ghahramani, Z. 2018. Probabilistic Deep Learning using Random Sum-Product Networks. *ArXiv*.

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

Rahman, T., and Gogate, V. 2016. Merging strategies for sum-product networks: From trees to graphs. In *UAI*.

Rahman, T.; Kothalkar, P.; and Gogate, V. 2014. Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of chow-liu trees. In *ECML-PKDD*, 630–645. Springer.

Rennie, J. D. M. 2005. Regularized logistic regression is strictly convex. Technical report, MIT.

Rooshenas, A., and Lowd, D. 2014. Learning sum-product networks with direct and indirect variable interactions. In *ICML*, 710–718.

Rooshenas, A., and Lowd, D. 2016. Discriminative structure learning of arithmetic circuits. In *AISTATS*, 1506–1514.

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *CoRR* abs/1708.07747.

Xu, J.; Zhang, Z.; Friedman, T.; Liang, Y.; and Van den Broeck, G. 2018. A semantic loss function for deep learning with symbolic knowledge. In *ICML*.