Target Curricula via Selection of Minimum Feature Sets: a Case Study in Boolean Networks

Shannon Fenn   SHANNON.FENN@NEWCASTLE.EDU.AU

Pablo Moscato   PABLO.MOSCATO@NEWCASTLE.EDU.AU
School of Electrical Engineering and Computing
University of Newcastle
University Drive
Callaghan NSW 2308
Australia

Abstract

We consider the effect of introducing a curriculum of targets when training Boolean models on supervised Multi Label Classification (MLC) problems. In particular, we consider how to order targets in the absence of prior knowledge, and how such a curriculum may be enforced when using meta-heuristics to train discrete non-linear models.

We show that hierarchical dependencies between targets can be exploited by enforcing an appropriate curriculum using hierarchical loss functions. On several multi-output circuit-inference problems with known target difficulties, Feedforward Boolean Networks (FBNs) trained with such a loss function achieve significantly lower out-of-sample error, up to 10% in some cases. This improvement increases as the loss places more emphasis on target order and is strongly correlated with an easy-to-hard curricula. We also demonstrate the same improvements on three real-world models and two Gene Regulatory Network (GRN) inference problems.

We posit a simple a-priori method for identifying an appropriate target order and estimating the strength of target relationships in Boolean MLCs. These methods use *intrinsic dimension* as a proxy for target difficulty, which is estimated using optimal solutions to a combinatorial optimisation problem known as the Minimum-Feature-Set (minFS) problem. We also demonstrate that the same generalisation gains can be achieved without providing any knowledge of target difficulty.

Keywords: Multi-Label Classification, Target Curriculum, Boolean Networks, k-Feature Set

1. Introduction

When students are first taught the concept of addition, they are not simply handed an eclectic set of many digit number pairs and their summations; rather, we provide them a learning curve of carefully chosen examples which increase in difficulty (i.e. number of digits). From standard classroom practices we can glean two key sources of inspiration regarding effective learning and teaching processes of humans: the curriculum of examples, and the curriculum of targets. This paper examines the latter.

Khan et al. (2011) found that humans naturally follow an easy-to-hard example-based curriculum when teaching a single-target concept to an unknown learner. Methods of scheduling examples this way have been explored in the machine learning community for some time, with positive results for non-convex scenarios (Bengio et al., 2009).

The typical explanation for the success of example curricula in non-convex scenarios is that it imposes a form of transfer learning where the simpler concept is discovered first and subsequently informs the more complex concept (Bengio et al., 2009). This leads neatly into a consideration of the second inspiration from the classroom: target curricula.

We consider the problem of learning discrete models for MLC problems with Boolean inputs, where closed-form solutions and gradient-guided optimisation techniques are unavailable. We use Feedforward Boolean Networks (FBNs)—defined in Section 1.2—as a case study for implementing target-curriculum regularisation with simple modifications to the typical L_1/L_2-Loss. We also introduce methods for estimating target overlap and target difficulty order in Boolean MLCs by finding a lower bound to the intrinsic dimension of the induced function. The intrinsic dimension has been presented as a proxy for the overall complexity of arbitrary data sets (Granata and Carnevale, 2016) suggesting its promise as a metric for ordering targets.

1.1 Contributions

We provide three novel contributions:

- we show that using hierarchical loss functions improves generalisation performance of FBNs trained on several circuit-inference problems possessing a natural target progression,

- we describe a simple method for estimating the complexity and overlap of Boolean targets by lower bounding their intrinsic dimensions using solutions to the minFS problem, and

- we demonstrate the success of the combined method on the same problem instances when knowledge of the target difficulty is withheld.

In the remainder of this section we define FBNs as a learning model, give background on MLC and outline existing work on sample and target curricula. In Section 2 we define hierarchical loss functions as well as the minFS problem and how it can be used to detect target overlap and estimate an appropriate curriculum. Finally, in Section 3 we present results on several problems.

1.2 Boolean Networks

This paper focuses on Boolean MLC problems where the targets possess differing complexities. In this section we provide definitions of the problem domain, its relevance and the representation of interest to this work: the Feedforward Boolean Network.

A Boolean MLC problem is one where inputs and targets take values in $\mathcal{B} \in \{0, 1\}$. An intuitive model for such a problem is a digital circuit, or FBN. A FBN with l inputs and m outputs computes a function of the form $f : \mathcal{B}^l \to \mathcal{B}^m$ by chaining computations from internal nodes (commonly called gates).
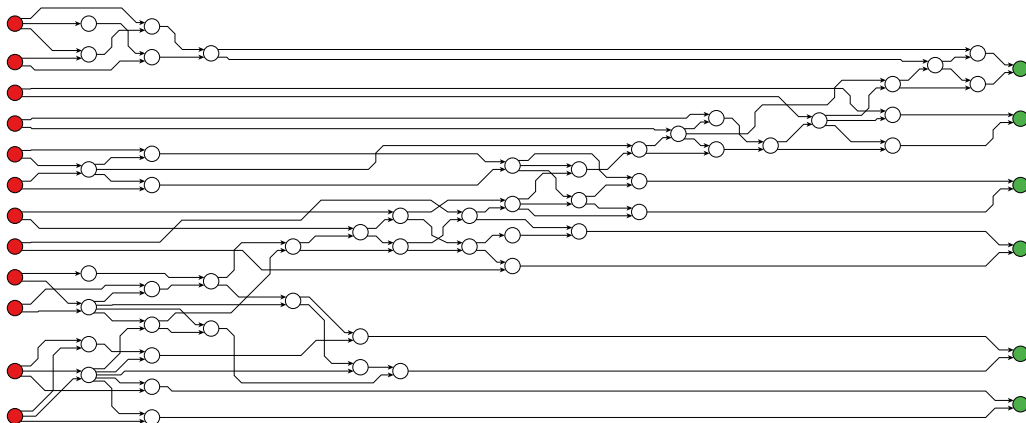
Figure 1: A 56-node FBN resulting from our learning procedure, which correctly implements the 6-bit addition function. Each node takes 2 inputs and computes the NAND function as its output. Inputs (far left) have been coloured red and outputs (far right) green. Note the ripple-carry style flow of information between sub-networks.

Internally, a FBN is a directed acyclic graph (DAG) where each node has an associated value in $\mathcal{B}$. Nodes take the values of their predecessors as input to a Boolean function which they provide as output. Input nodes are those with no predecessors and their value is provided from outside the network. The output of the network is simply the values at a particular subset of externally visible output nodes. A FBN can thus compute a function from many inputs to many outputs by combining simpler functional elements. Figure 1 shows an example FBN which correctly implements a 6-bit adder.

One advantage to working with this representation is that two singleton functionally complete operator sets exist: the negated-and (NAND) and negated-or (NOR) functions. Every Boolean mapping can be computed by some degree-2 DAG consisting purely of NAND nodes. We will use this representation for the remainder of the paper.

FBNs have a number of other advantages as learning models. They are efficient in time and memory to simulate and trivial for direct hardware implementation and logical rule conversion. Logical relationships are simple for humans to extract information from which is important if the goal of the learning procedure is knowledge-generation. While other representations have received significantly more attention, FBNs are trivial to map to existing digital hardware—an open problem for neural nets. They can use packed integer representations and bitwise arithmetic to improve model evaluation speed during training and deployment. Recent literature presents methods for binarising the internal layers of deep neural nets precisely for these reasons (Courbariaux et al., 2016).

As the name suggests, FBNs compute with purely Boolean logic, rather than using weighted sums and continuous activations. This brings challenges, the most prevalent being the lack of gradient-based optimisation procedures or closed form solutions. All optimisation techniques discussed below use general purpose combinatorial meta-heuristics, such as Simulated Annealing (SA). Herein all networks were optimised using local search in the space of graph structures—possible due to the functional completeness of the NAND operator.

3

1.3 Learning with Boolean Networks

Since the late publication of Turing's first description of a machine which learns by changing the connections of a NAND-gate circuit (Turing, 1948) the majority of the work on learning with a Boolean Network model has been done in the field of physics. Some time later, Patarnello and Carnevali (1987), and subsequently others (Van den Broeck and Kawai, 1990; Goudarzi et al., 2014), demonstrated that FBNs could generalise well on some problems, despite training set sizes significantly smaller than the space of all possible patterns.

Learning FBNs is a *combinatorial* optimisation problem, typically solved using meta-heuristics such as SA or genetic algorithms. Patarnello and Carnevali (1987) optimised networks by SA in the space of feedforward structures. Each move in the search procedure consisted of a random change in a single connection. Moves which also changed the node activation have been considered (Van den Broeck and Kawai, 1990; Goudarzi et al., 2014), however in light of the NAND function's functional completeness, this becomes unnecessary.

To simplify the following loss function definitions we define the $n \times m$ error matrix E, which fully characterises the mistakes made by a candidate network on a set of n example patterns with m targets. E is the element-wise absolute difference between the network output matrix Y' and the target matrix Y, with elements given by

$$E_{i,j} = \left| Y_{i,j} - Y'_{i,j} \right|,$$

where $E_{i,j} \in \mathcal{B}$ is the error for the jth target for the ith example.

Even for problems with multiple outputs, the only loss functions used to date (Patarnello and Carnevali, 1987; Teuscher et al., 2007; Goudarzi et al., 2014) have been analogues of the L_1-loss:[1]

$$L_1 = \frac{1}{m \, |\mathcal{I}|} \sum_{j=1}^{m} \sum_{i \in \mathcal{I}} E_{i,j},$$

where $\mathcal{I}$ is the set of example indices shown to the network. This loss treats all examples and targets equally and is the natural first choice for a guiding function, but for multi-target problems there are other options which we explore in Section 2.1.

So far no work has considered differences between learning single and multiple output FBNs. Results for Multi Task Learning (MTL) suggest that, with multiple related target, some networks may be favoured due to target correlation, increasing generalisation performance. To explore the possibility of further improvements, we use FBNs as a case study in the effects of target curriculum enforcement by hierarchical loss-functions on purely discrete models. In the following section we outline the relevant work on target curricula.

1.4 Example Curricula

The approach we present in this work draws strong inspiration from *example* curricula. In this subsection we give a brief outline of some relevant work to better frame our approach.

Curriculum learning by applying a ranking of *examples* can be split into two main camps: Curriculum Learning (CL) and Self-Paced Learning (SPL) (Kumar et al., 2010). In the former the ranking is determined a priori while in the latter it is learned jointly

---

1. With purely binary predictors and targets the L_1- and L_2-loss are identical.

with the model. Jiang et al. (2015) demonstrated a combined approach—intuitively called Self-Paced Curriculum Learning—where constraints on the example-ranking are imposed by a fixed curriculum and the ranking is learned jointly with the model but subject to the curriculum constraints. SPL and CL emerge as special cases of their framework. They present positive results in comparison to baseline methods on matrix factorization and multimedia event detection problems.

Spitkovsky et al. (2009) demonstrated the success of a curriculum of samples of *increasing complexity* on the problem of determining grammar from free-form sentences. Their proxy for complexity in this case is the length of the example sentence. They also noted the appearance of a "sweet-spot" in sentence length, the inclusion of samples above which reduced performance. The combination of both their curriculum and complexity limitation improved on the state-of-the-art.

It is important to note that the terms Curriculum Learning (CL) and Self-Paced Learning (SPL) used within this subsection related *example* curricula. It is possible to also consider a learner—in a MLC framework—to be following a curriculum of *targets* (or to be self-paced in the same respect). We primarily cover the former case of a fixed curriculum of targets, and throughout this paper "curriculum" will refer to a curriculum of targets.

1.5 Target Curricula

Learning multiple targets concurrently with a shared representation has been shown, theoretically and experimentally, to improve generalisation results when there are relationships between the targets (Caruana, 1997; Wang et al., 2016). The number of publications discussing multi-label and multi-task learning is growing rapidly and using a *curriculum* of targets, while less general, is also beginning to garner attention.

The importance of sharing knowledge between targets increases when there is a disparity in target complexity. Gülçehre and Bengio (2016) showed a visual task, on which many popular methods—including deep neural nets, SVMs and decision trees—failed. This same task became solvable when a simpler hinting target was provided (in much the same way as introduced by Abu-Mostafa, 1990). However, unless relative target difficulties are known (or suspected), we also face the issue of determining an appropriate order during learning.

Pentina et al. (2015) presented a self-paced method using adaptive linear SVMs. The implementation was model-specific, providing information transfer by using the optimal weight vector for the most recently learned task to regularise the next task's weight vector. They discover the curricula by solving all remaining tasks and selecting the best fitting, resulting in a quadratic increase in training time over solving tasks individually. While this may be acceptable for a linear model, it would be prohibitive for a FBN.

Li et al. (2017) implemented task curricula with linear models by weighting the contribution of each task-example pair to the loss function. They used a regulariser in which task-example pairs with low training error receive larger values for these weights. However, this approach does not appear applicable to inferring highly non-linear discrete models.

Lad et al. (2009) present the only model-agnostic approach to determine an optimal order of tasks that we are aware of. They use conditional entropy to produce a pairwise order matrix and solve the resulting NP-Hard Linear Ordering Problem (see Ceberio et al., 2015, for a definition). Their approach is not applicable to Boolean MLCs since the conditional

entropy difference calculation they use reduces to the difference of the entropies of the individual targets; which, for balanced binary targets, rapidly approaches zero as more training samples are given.

The impact of target order is highlighted by Read et al. (2011). Their approach uses *classifier chains*: independent learners for each binary label, trained in an arbitrary order, with the output of all prior classifiers included as input to each subsequent classifier. The authors note that the ordering of targets has a notable effect on classifier performance; which they address using ensembles of chains, each with a different random order. They note consistent superiority of the ensemble approach over a single random ordering, which suggests non-trivial variability in the impact of different orderings. Due to the large space of permutations, a principled approach to determine appropriate orderings a priori would be of value.

We have not seen work using hierarchical loss-functions (or similar regularisers) to impose a target curriculum, nor have we seen methods for determining a curriculum by use of feature selection methods. The following section describes our approach.

2. Learning Targets in Order

Here we describe three loss functions which we used to enforce a curriculum of targets, as well as a novel method for obtaining said curriculum which reduces the problem to m instances of the Minimum-Feature-Set (minFS) problem.

2.1 Hierarchical Loss Functions

Assume a two-target Boolean MLC for which we have three candidate learners, and assume we know that the first target is "easier" under some definition than the second. The first learner labels target 1 correctly and target 2 incorrectly for all examples. The second learner does the opposite, and the third learner labels 50% of the examples incorrectly for both targets. All three networks will have an equal $L1$-loss.

The rationale behind the loss functions we are about to define is that the first of the above learners should be preferred. By focusing on easier targets earlier in the training process the network is more likely to find substructures useful to later targets. The expectation is faster convergence speed, better generalisation or both.

The L_1-loss is simply the mean of the error matrix. In addition, we consider three loss functions which hierarchically aggregate the error matrix elements in progressively more strict ways:

- a linearly weighted mean: L_w,

- a locally hierarchical mean: L_{lh}, and

- a globally hierarchical mean: L_{gh}.

All definitions below assume the targets are ordered by difficulty, with the simplest target at the lowest index.

The first function, L_w, encourages the target curriculum by weighting the easier targets more highly. The weighting is linear and the resulting summation is normalised to the unit

interval:

$$L_w = \frac{2}{m\,(m-1)\,|\mathcal{I}|} \sum_{i \in \mathcal{I}} \sum_{j=1}^{m} (m - j + 1)\, E_{i,j} \ .$$

We named the second function, L_{lh}, the *locally hierarchical* loss, as it enforces a hierarchy of targets on a per-example basis. The learner is rewarded for correctly labelling target j of example i if and only if it has also correctly labelled all prior targets $\{1, 2, ..., j - 1\}$ *for that example.* To define this function, for each example i, we define the recurrence:

$$a_{i,1} = E_{i,1}$$

$$a_{i,k+1} = \begin{cases} E_{i,k+1}, & \text{if } a_{i,k} = 0 \\ 1, & \text{if } a_{i,k} = 1 \end{cases} ,$$

from which we can define

$$L_{lh} = \frac{1}{m\,|\mathcal{I}|} \sum_{i \in \mathcal{I}} \sum_{k=1}^{m} a_{i,k} \ .$$

The final loss, L_{gh}, follows the same principle as the former but across all examples. For this reason we named it the *globally hierarchical* loss. In this case the learner is given reward for correctly labelling target j (of any example) if and only if it has also correctly labelled all prior targets $\{1, 2, ..., j - 1\}$ for all examples given. This function is effectively a "soft" equivalent to learning the targets rigidly in order. We can define L_{gh} using the mean per-target errors (the row-means of E):

$$\delta_j = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} E_{i,j} \ .$$

These can be used to define another recurrence,

$$b_0 = \delta_0$$

$$b_{k+1} = \begin{cases} \delta_{k+1}, & \text{if } b_k = 0 \\ 1, & \text{if } b_k > 0 \end{cases} ,$$

giving

$$L_{gh} = \frac{1}{m} \sum_{j=1}^{m} b_j \ .$$

Unless the difficulty order is known, or suspected, from domain specific knowledge we also require a method for estimating the target order from the training examples. We describe a method suitable for Boolean or categorical MLCs in the following section.

2.2 Measuring Target Overlap and Difficulty Order

The key requirement for ordering targets is an estimation of their relative difficulty. It is likely most suitable to use domain knowledge where available, however in cases where this is impossible or difficult to obtain we suggest a rule of thumb: order by increasing intrinsic dimension. We present an intuitive approach below using a combinatorial problem which

did not significantly increase the learning time for the instances considered. In Section 3.4 we show that the target order estimated by this method gives generalisation improvements close to those obtained using the known order (Section 3.3).

Our key assumption is that a target can be expected to be simpler than another if its intrinsic dimension (the number of input variables on which it truly depends) is smaller. This assumption is strong but there is also a strong basis for it as a rule-of-thumb in the Boolean domain, as a function's intrinsic dimension places an upper bound on the size of the smallest FBN which implements it. Furthermore, intrinsic dimension is showing promise as a method for estimating the complexity of arbitrary data (Granata and Carnevale, 2016) and as such we expect the validity of this idea may extend beyond the Boolean case.

The likelihood of effective information transfer can also be estimated by measuring the overlap in intrinsic input features. Thus estimating both target overlap, and relative complexity, is reduced to the problem of determining the minimum set of input features on which a valid function can be defined; that is, where no pattern is mapped to multiple target values. This is well known as the Minimum-Feature-Set (minFS) problem.

2.2.1 THE MINIMUM-FEATURE-SET PROBLEM

The smallest intrinsic dimension of any Boolean function which is in full agreement with given set of examples can be found using the Minimum-Feature-Set (minFS) problem. It asks: what is the smallest subset of given input features, on which a single target can still form a non-contradictory mapping? This problem, and particularly this view of it, is not in any way new to the machine learning community (Davies and Russell, 1994). However, to our knowledge, it has not been used to estimate relative target difficulty before.

This equivalent decision problem is known as the *k-Feature-Set (kFS)* problem. Formally:

k-Feature-Set

Given: A binary $n \times p$ matrix, $\mathcal{M}$, where the rows describe n examples and the columns describe p input features as well as n-element target vector, $\mathcal{T}$, and a positive integer k.

Question: $\exists S \subseteq [1, ..., p], |S| \leq k$, such that, $\forall i, j \in [1, ..., n]$ where $\mathcal{T}_i \neq \mathcal{T}_j$, $\exists l \in S$ such that $\mathcal{M}_{i,l} \neq \mathcal{M}_{j,l}$?

That is: is there a cardinality k subset, S, of the input features, such that no pair of examples which have identical values across all features in S have a different value for the target feature?

The problem is NP-complete (Davies and Russell, 1994) and assumed to not be fixed-parameter tractable, when parametrised by k, under current complexity assumptions (Cotta and Moscato, 2003). Nonetheless the exact solver used herein solved all instances in significantly less time than was required for subsequently optimising the FBN. At least one fast meta-heuristic solver exists for a generalised version of the problem, with reasonable run-times for very large instances (Rocha de Paula et al., 2015), and problem instances can be reformulated as instances of the well known Set Cover Problem, for which there are numerous powerful heuristic solvers.

We implemented instances of the minFS problem below as Mixed-Integer Programs and solved using them using the Python interface to the IBM CPLEX solver. With a solution

procedure in hand we can now define a simple method for estimating the target difficulty order and the level of target overlap.

2.2.2 Target ordering method

Given a procedure for solving the minFS problem our proposed method for ordering the targets reduces to solving m such instances; one for each target. For all instances the input feature matrix $\mathcal{M}$ is identical to the entire input matrix for the original learning problem. Each instance is solved to optimality resulting in a set of features $S_1, ..., S_m$ for each target.

An estimate of the intrinsic dimension for a target i is then simply $|S_i|$. Targets are sorted by this estimate with ties broken randomly, and the resulting total order is used in the loss functions defined in Section 2.1. These losses could be redefined to treat a subset of targets equally to avoid tie-breaking but we did not explore this possibility.

2.2.3 Target overlap score

Estimating target overlap also follows from the minFS solutions. For two targets, y_i and y_j, with solutions, S_i and S_j, the target overlap is estimated by the Szymkiewicz-Simpson overlap coefficient:

$$\sigma(i,j) = \frac{|S_i \cap S_j|}{min(|S_i|, |S_j|)} \ .$$

The overlap coefficient gives a value between 0 and 1 where 0 indicates that there is zero overlap between two sets and 1 indicates that one set is a subset of the other; values in between occur when sets are partially overlapping. This measure was chosen over the more popular Jaccard index as we wish to consider all fully overlapping cases ($A \subseteq B$) equivalent regardless of their relative sizes.

We can then define the *nestedness* score, η, as the mean overlap coefficient between successive targets (when ordered by the method in Section 2.2.2) giving the formula

$$\eta = \frac{1}{m-1} \sum_{i=2}^{m} \sigma(i, i-1) \ ,$$

which takes the value 1 when the minFS solutions for all targets forms a perfectly nested sequence of sets, and the value 0 when the sets have no pairwise overlap. A high value for η is thus a necessary but not sufficient condition for strongly inter-dependent Boolean targets. A value near zero indicates that there is unlikely to be any benefit gained from a target curriculum.

The nestedness score allows us to rule out some Boolean MLC problems for which a target curriculum would be inappropriate, and the proposed target ordering method gives us a curriculum otherwise. We show experimental results in the following section.

3. Experimental Results

It is important to see how the loss functions defined in Section 2.1 affect generalisation performance over a range of training set sizes. When optimising L_1, generalisation performance for all problems considered transitions from essentially random behaviour to near perfect

9

prediction as training set size increases. With this in mind we generated 1000 samples each for a range of training set sizes chosen to span that transition by a reasonable margin. We report the training set size as the fraction, $s = |\mathcal{I}|/2^l$, of all possible patterns.

For each sample we trained one network for each loss function until they achieved zero training error. This allows us to report the difference in test error between the network optimised under L_1 and each of the remaining losses, for the same training set. Using the same training set for each loss removes the significant variation in instance difficulty as a confounding factor.

3.1 Data Sets

Here we define the problems considered. We first outline problems for which we already have a difficulty curricula. Then we consider several published models of real-world phenomena and finally we test our overall approach on the problem of discovering regulatory network update rules from discretised time-series data.[2]

3.1.1 Circuit inference test-beds

For the first test-bed we chose five circuit-inference problems: binary addition (add) and subtraction (sub) and cascaded variants of three typical circuit-learning and genetic programming problems: multiplexer (cmux), majority (cmaj) and parity (cpar). These were chosen as they all possess a natural hierarchy of targets.

Binary addition and subtraction are well known multi-output problems. For the three remaining problems, multi-output versions are constructed by cascading single-output sub-circuits; each successive output is the result of applying the original function over a larger set of inputs. By this method we construct problems for which we know there is a difficulty order, and what that order is. More detailed definitions of these problems are given in Online Appendix 1.

The cascaded parity and multiplexer possess the strongest dependencies between successive targets, as each subsequent output can be computed directly from the previous output and an additional subset of inputs not already used. The cascaded majority, binary addition and binary subtraction problems have weaker dependencies, requiring extra inputs as well as the previous output and new inputs to compute the next output. This allows us to see the effect of imposing target curricula of problems with varying levels of inter-target dependency.

FBNs are a high variance learner. As such we selected problem sizes so that a meaningful effect size can be seen, but also so that 1000 instances per loss/sample-size combination could be completed in under a week. Table 1 outlines the dimensions selected, as well as the ranges in training sample size.

The hierarchy of targets is known for the above problems since they are constructed from cascading well-known circuits. This ground-truth allows us to further evaluate the methods outlined in Section 2.2 by considering the correlation between this known order and the predicted order.

---

2. Data and code are available at
   `github.com/shannonfenn/Multi-Label-Curricula-via-Minimum-Feature-Selection`.

Problem	Inputs	Targets	Training set size	Example pool size
Cascaded Parity	7	7	[8, 112]	128
Cascaded Majority	9	5	[8, 384]	512
Binary Subtraction	10	5	[8, 192]	1,024
Binary Addition	12	6	[8, 208]	4,096
Cascaded Multiplexer	15	7	[8, 180]	32,768
74181 ALU	14	8	[64, 384]	16,384
FA/BRCA pathway	28	28	[8, 96]	1,024
Mammalian cell cycle	10	9	[8, 180]	1,024
Yeast	10	9	8	9
E. Coli	10	8	4	5

Table 1: Instance sizes of the test-bed problems.

3.1.2 Boolean Models of Real Phenomena

In addition to the problems above we also conducted similar experiments using real-world models. This included a commercial integrated circuit: the 74181 4-bit Arithmetic Logic Unit (ALU), and two published biological models.

The 74181 ALU has 14 inputs and 8 outputs. It performs 32 different arithmetic and logical operations on an 8-bit input with carry in, generating a 4-bit output with carry out as well as 2 other carry-related outputs useful for faster calculations when chaining multiple ALUs. The 74181 represents a reverse engineering problem of a realistic size that is less idealised than the above problems, as there is no expected absolute ordering to the targets.

The biological models we considered were Boolean models of the Fanconi Anemia/Breast Cancer (FA/BRCA) pathway and the mammalian cell cycle (Poret and Boissel, 2014). Boolean regulatory network models like these describe the time-evolution of gene activity, levels of regulatory products and protein activity with a single Boolean variable per element. The system dynamics are then defined with Boolean update functions and an update method (synchronous, partly or fully asynchronous). Both models we consider are synchronous as this enables the inference of a single deterministic update equation by treating successive time steps as input-output pairs. More detail on all models in this section are given in Online Appendix 1.

For these three problems we ran experiments just as in Section 3.1, with the exception of the ALU problem for which we only trained 100 networks per training set size due to time constraints. Similarly, due to the large memory requirement of generating all 2^{28} patterns for the FA/BRCA model, we used a reduced example pool of 2^{10} patterns sampled uniformly. The problem dimensions, training set sizes and example pool sizes are given in Table 1. We have no suspected optimal target order for these problems, so the ordering used for all ordered losses was found using the minFS-based intrinsic dimension estimation described in Section 2.2.

3.1.3 Learning Regulatory Networks From Time-Series Data

While conducting the above experiments we observed that the primary performance gain is seen at smaller sample sizes. For this reason we chose a real-world test-bed where a key issue is the extremely limited amount of available data: inferring Boolean GRN update models from time-series gene expression data. Barman and Kwon (2017) provide binarised time-series data for an *E. coli* GRN and a cell-cycle regulatory network of fission yeast: each consisting of 10 nodes and only a handful of examples. One desirable result is a synchronous update model such as in the models described in Section 3.1.2, and one way to achieve this is to treat each sequential pair of states as an input-output pair thus deriving a FBN which predicts the next state from the current state. Again, further detail is given in Online Appendix 1.

Some preprocessing of the data were required. First we removed repeat patterns—these occur due to time disparities which we are not attempting to model. Instead we require pairs of distinct patterns as otherwise the requirements for a valid minFS instance are not met. Secondly we removed constant targets since they represent nodes for which there is no information suggesting a relationship to any other nodes: one such target was removed from the yeast data, and two from the *E. coli* data.

Finally, since we have very small example sets allowing rapid training, we use leave-one-out cross validation for all patterns whose removal does not introduce additional constant targets (the first was the only such pattern in both cases). For each fold we learned 1000 networks for each configuration of interest, to account for the significant variance typical in such heavily under-determined problems.

Our initial tests including all targets of the yeast and *E. coli* data sets as a single problem showed little improvement from the ordered losses. Preliminary feature set analysis suggested that there were distinct, shallow hierarchies among the targets (see Figure 2), rather than the deeper hierarchies seen in prior problems. A lack of improvement is unsurprising in this case since imposing an order upon unrelated targets is not expected to offer any benefit and to potentially detriment the performance.

Due to this we opted to learn networks only for the nodes involved in a hierarchy (2 for yeast and 1 for *E. coli*). For fair comparison we compared with the baseline L_1 network on both target sets: only those in the suspected hierarchy and all targets. Results are only shown for the former however, since the results of learning a single overall network using L_1 were notably worse. Again we used estimated target curricula for all ordered losses (see Section 2.2).

3.2 The Learning Procedure

Here we describe the local search procedure used for training. We learned each function by optimising FBNs comprised of only NAND gates using stochastic local search. As the NAND operator is functionally complete the optimisation process does not need to consider node transfer functions and can instead learn any mapping by modifying just the network structure. We set network size, n_g, for all problems at $21m$ and represented the structure using a $2 \times n_g$ integer matrix, with column i containing the indices of the two sources for the ith node.

G2
{G7}

G8
{G7}

SK
{start}

PP
{Slp1}

G6
{G4, G7}

Ste9
{start, PP}

Rum1
{start}

Cdc2/13
{Slp1, Ste9}

Cdc2/13*
{Slp1, Wee1/Mik1}
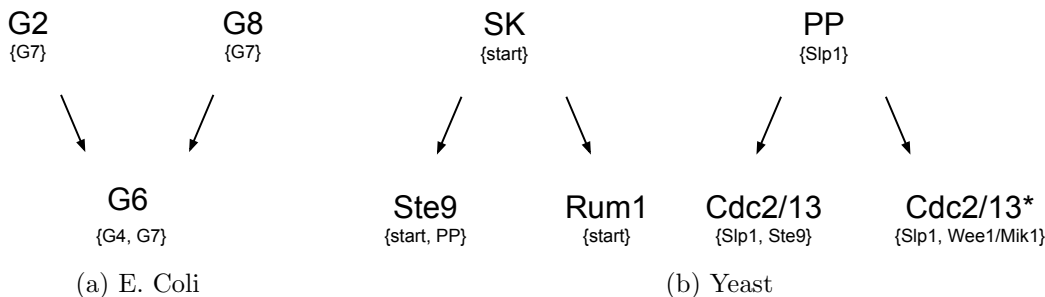
(a) E. Coli

(b) Yeast

Figure 2: Possible target hierarchies found using minFS in the yeast and *E. coli* data sets. In the *E. coli* data (a) we see the target G6 (with feature set $\{G4, G7\}$) may benefit from being learned after G2 or G8 (both have the feature set $\{G7\}$). In the yeast data (b) we see two potential hierarchies: either of Ste9 or Rum1 (both with feature set $\{start, PP\}$) may benefit from being learned after SK, and either of Cdc2/Cdc13 or Cdc2/Cdc13* (with feature sets $\{Ste9, Slp1\}$ and $\{Slp1, Wee1/Mik1\}$ respectively) may benefit from first learning PP.

We ensure the network remains feedforward by imposing a topological ordering of nodes; each node may only accept input from nodes before it in the ordering. Since any DAG has at least one topological ordering, no structure is excluded by this constraint.

Training involved performing stochastic local search in the space of valid feed-forward structure, using the Late-Acceptance Hill Climbing (LAHC) meta-heuristic (Burke and Bykov, 2008) with random restarts. This method assumes a black-box view of the learner and is similar in implementation to SA while being less sensitive to scaling in the guiding function and requiring only a single meta-parameter in place of a cooling schedule. While the particulars of the optimiser are not of relevance to this work, a brief description is given below and the pseudo-code is given in Online Appendix 1.

LAHC works similarly to typical stochastic hill climbing: a modification is made to the current solution and accepted based on some criterion. This modification involves a random change to the source of one connection in the DAG. A modification is accepted if the resulting network has equal or lower loss than the solution obtained L iterations beforehand.

The single meta-parameter, L, is the length of a history of costs which provides a dynamic error bound. In preliminary tests, the out-of-sample error was consistent over a wide range in L and so we selected the value which provided the best average convergence speed when using the L1-loss. The resulting value was 250 for all problems, except parity, for which it was 1000.

For MLCs with discrete inputs there are only finitely many possible patterns. Since our test-bed problems are fully defined, we construct each test set from all possible patterns except those used in that particular training set. In doing so the true out-of-sample error for each sample is known.

3.3 Results for Hierarchical Loss Functions

In this section we present results showing that the losses defined in Section 2.1 improve the out-of-sample error achieved on all test-bed problems mentioned in Section 3.1.

Learning FBNs with structure-based local search displays high variance. As such, each point shown in Figures 3, 4, and 5 represents the mean performance over 1000 different training sets of that size. We have also displayed 95% confidence intervals as transparent bands in Figure 4. For Figure 3 this interval was too small for bars or bands to be visible.

The overall test accuracy as it varies with sample size for the 7-bit cascaded parity problem is shown in the top of Figure 3. A phase transition between poor and near perfect generalisation is expected (Patarnello and Carnevali, 1987). What we see in Figure 3 is a leftward shift in this transition as we impose an increasingly strict order to the targets. Less examples are required to achieve the same generalisation if a difficulty-based target curriculum is imposed.

Even for a single problem instance the improvement is not expected to be consistent across all targets. In fact our earlier reasoning suggests the improvement should increase for successive targets. In the bottom of Figure 3 we have also shown the test accuracy improvement given by each loss, for each target of 7-bit cascaded parity. The first two targets are not shown; as expected there is little improvement for them. Beyond this, we see that the improvement increases with each successive target until the last (the same trend appears in all five problems). This peaks at the second-last target with an increase from 58% test accuracy using L_1, to 84% test accuracy using L_{gh}.

Generalisation improvements can be most readily observed when looking at the test accuracy differences between L_1 and each of L_w, L_{lh} and L_{gh} as they vary with s. Figure 4 shows the mean of these differences with respect to s along with transparent bands indicating the 95% confidence interval of the mean. We can see that the improvement differs significantly between targets but is consistently positive and statistically significant across a range of training set sizes for all five problem instances.

More importantly, the loss which most strictly enforces the learning order: L_{gh}, also confers the largest improvement, and vice versa for the loss which least strictly enforces it: L_w. This phenomena is consistent across all five problems and confirms the central idea presented in this paper.

In terms of overall generalisation accuracy we can see that training to a curriculum of targets offers a significant advantage in Boolean MLCs with target hierarchies. Furthermore we can see that a curriculum can be enforced in a combinatorial black-box optimiser using only slight modifications to the typical guiding function.

3.4 Evaluating the Discovered Curricula

Here we evaluate the effectiveness of the proposed methods for estimating target overlap and difficulty order described in Section 2 on the same test-bed problems. We also observe how the improvement conferred by hierarchical loss functions varies when random orderings are used.
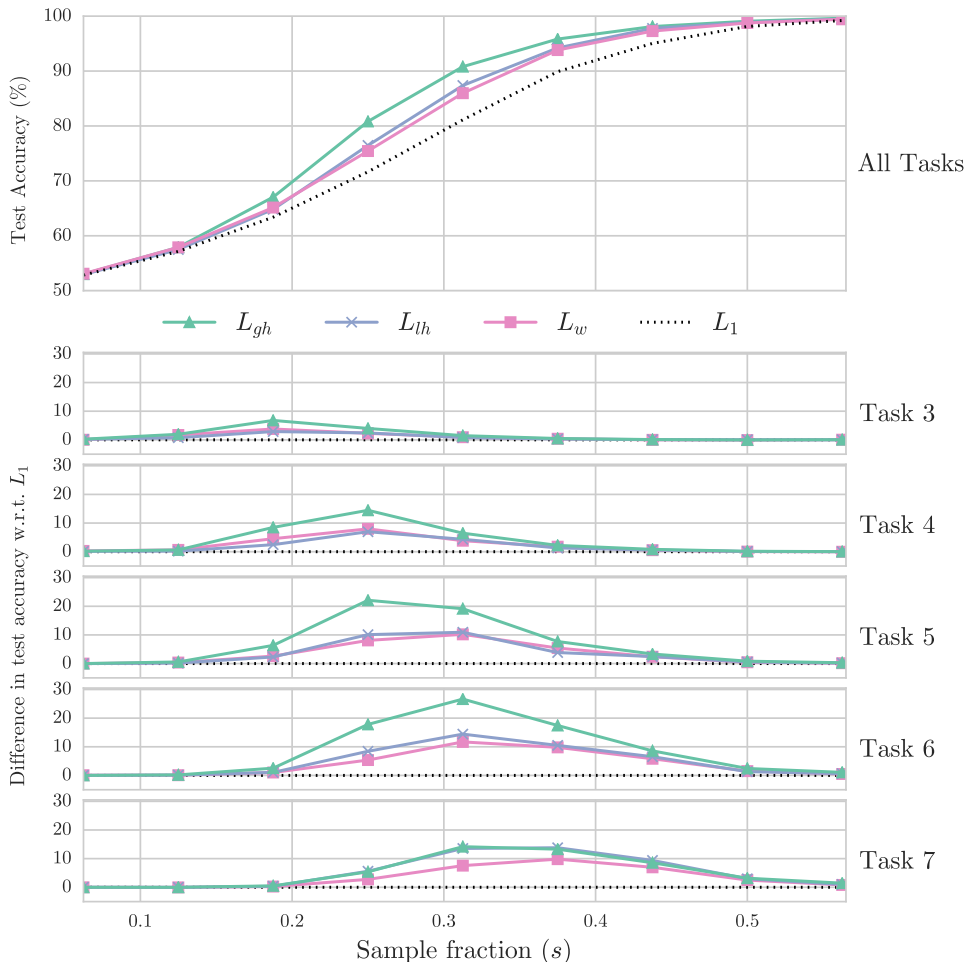
Figure 3: Generalisation performance of each loss function on the 7-bit cascaded parity problem. The top plot shows the mean test accuracy averaged across all targets for each loss while the remaining plots show the mean difference in test accuracy, on each target, between L_1 (dotted baseline) and each of L_w, L_{lh} and L_{gh}. The first two targets are not shown as they show only a negligible improvement. Confidence intervals (95%) are also not shown as they were too narrow to be visible, the widest being less than 2%. A leftward shift in the transition (top) from poor to near-perfect performance can be seen. This becomes more pronounced as the loss more strictly enforces the learning order ($L_w \rightarrow L_{lh} \rightarrow L_{gh}$) with L_{gh} achieving a 16% reduction in the number of training examples needed to achieve above 90% mean test accuracy. The differences in test accuracy for individual targets (bottom) shows an increase in improvement with target difficulty, as expected; reaching a large peak improvement—from 58% to 84% test accuracy—for the sixth target. On this particular problem the difference between L_{lh} and L_w is not statistically significant. For all loss functions the out-of-sample accuracy is significantly better than obtained with L_1 (dotted line).
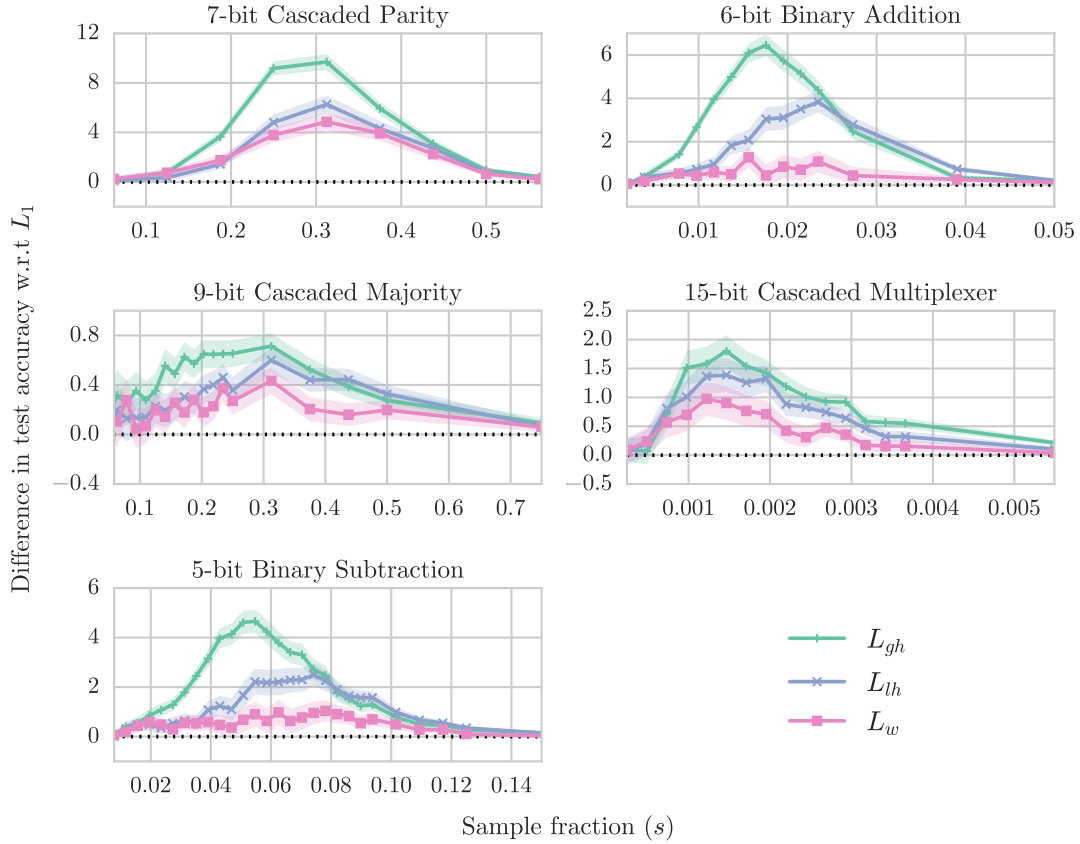
Figure 4: Mean difference in test accuracy between L_1 (dotted baseline) and each of L_w, L_{lh} and L_{gh} on each training sample and problem instance. The 95% confidence interval of the mean is shown with transparent bands. On all problems we can see a statistically significant net increase in test accuracy—though the magnitude varies widely between problems. We can also see that this improvement increases as the loss more strictly enforces the ordering ($L_w \rightarrow L_{lh} \rightarrow L_{gh}$).

3.4.1 Effectiveness of order estimation

We know the difficulty curricula for the problems in Section 3.1.1. Thus we can quantify the effectiveness of the proposed target ordering method on these, by using the rank correlation between the known and estimated orderings. Since these are permutations there are no ties to consider and we can use the simplest version of Kendall's τ:

$$\tau = \frac{P - Q}{P + Q} ,$$

where P is the number of target pairs which are placed in the same relative order under both orderings, and Q the number which are not. Being a correlation, τ takes values in $[-1, 1]$, with 1 indicating the two orderings are identical, 0 indicating they are uncorrelated and -1 that they are inverses.

We used the same experimental configuration as before but for each training instance we randomly shuffled the targets, to reduce the effect of any deterministic biases. Then we estimated an ordering using the minFS-based method and computed the rank correlation between this ordering and the order used in Section 3.3, as well as the nestedness score, η, defined in Section 2.2.3. Finally we learned one network for each instance using L_{gh} with the estimated target order. Only L_{gh} was considered since it produced the largest improvement with the known order (see Figure 4). These results are shown in Figure 5.

For each problem, Figure 5 shows the nestedness score and the rank correlation between the known and automatically discovered target orders; as well as the generalisation improvement imparted by L_{gh} when given each ordering.

We see that not all problems are easily detected as possessing overlapping hierarchical targets. On parity and majority the mean nestedness score and τ quickly reach 1.0 indicating that the proposed method is successful at automatically determining that there is an overlap, and the correct order of targets. For addition and subtraction the results are weaker but still promising however for the multiplexer there is only a weak correlation between the expected and known target orders and a low nestedness score. It is interesting that, even with a weakly correlated target order, some improvement is still gained from enforcing that order.

Except for CMUX—for which there is still an overall improvement—the third row in Figure 5 shows a high agreement between test accuracies gains using given and automatically detected target orderings. With zero prior knowledge of the respective target difficulties, the minFS-based method combined only with a slight change in the loss function, yields significant improvements in generalisation. This is promising given we are considering a learner which was already leveraging the expected benefits associated with a shared internal representation.

3.4.2 How performance correlates with target order

The next experiment we describe had two principal aims: to examine the relationship between the performance of hierarchical loss functions and the given curriculum, and to determine if the difficulty based curriculum that we have assumed as ground truth is actually optimal.

To do this we used a slightly smaller single problem—5-bit addition—and trained networks using L_{gh} with randomly generated target permutations and L_1. This gives us another
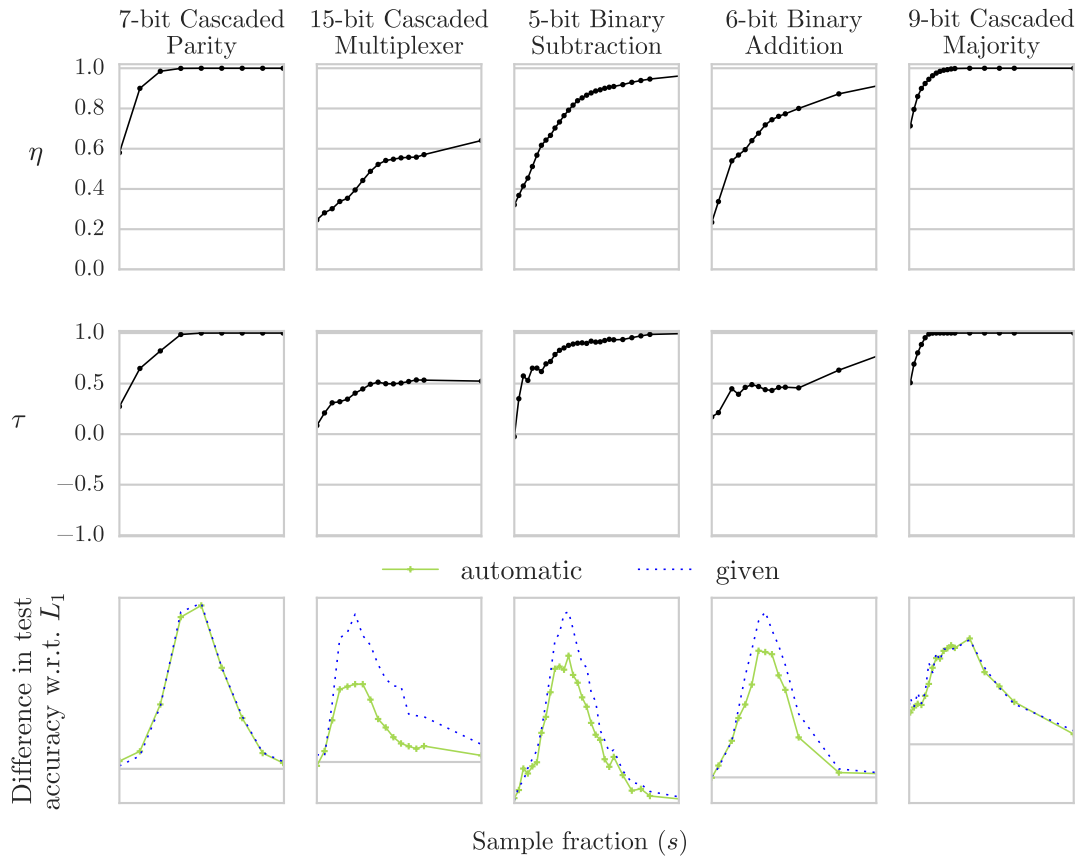
Figure 5: Mean increase in test accuracy—over networks trained using L_1—for networks trained using L_{gh} with the known target order (given) and the automatically detected target order (automatic). For all problems except CMUX both τ and η reach values above 0.5 before the respective peaks in test error difference. For these same problems the detected target order provides the nearly the same benefit as the known target order, which is no surprise given the strong correlation between the known and estimated orderings. The problem with the largest deviation, CMUX, is also the one for which τ and η are weakest. Note: the x-axes are only shared within columns, and the y-axis for the third row is not shared, since some problems have a much larger improvement than others.

important baseline—random target orderings—and also allows us to see how the performance of L_{gh} varies with τ, and if the least-to-most-significant curriculum we propose is outperformed by some other ordering.

We did not draw permutations uniformly since the rank correlation, τ, is heavily skewed over the space of permutations; the values ± 1 are only represented by 2, in combinatorially many, possibilities. However for a n-target permutation there are only $\lceil n \times (n-1)/2 \rceil$ unique values of τ, so selecting permutations uniformly distributed over τ is trivial. As such we trained 5 networks for each of 50 permutations (selected with replacement) for every value of τ over the range of sample sizes. The baseline L_1 does not vary with order so we again learned 1000 networks per sample size.

Figure 6 shows the resulting difference in test accuracy. Each line shows the mean improvement (over networks trained using L_1) as it varies with sample size for networks trained with L_{gh} for all permutations with that correlation value. One clear observation is that permutations which positively correlate (blue) with the easy-to-hard ordering give improvement while permutations which negatively correlate (red) actually detriment performance. The second key observation is that this varies with the strength of the correlation; and, most importantly, the largest improvement is given by the exact easy-to-hard curriculum ($\tau = +1$).

These results confirm our intuition that an easy-to-hard ordering is optimal—at least for this method of enforcing a target curriculum. They also agree with the results seen in Section 3.4.1 which showed that some improvement was seen for the CMUX problem even with a weak mean order correlation, but also that weaker correlation in the discovered curriculum reduced the improvement given by L_{gh}.

3.5 Real-World Problems

Here we present results for the experiments described in Sections 3.1.2 and 3.1.3. We lack an expected target order so the results for all ordered losses are with automatically detected curriculum (Section 2.2).

For the ALU, mammalian cell-cycle, and FA/BRCA pathway models we have the same regime as for the prior test-beds, so we have presented the results in the same manner. Figure 7 shows the mean difference in test accuracy between each hierarchical loss function and the baseline L_1 as sample size is varied. We also see that with the mammalian cell-cycle model there is much less differentiation between the hierarchical losses, in fact L_w appears to have given the most benefit. Overall, in all three cases imposing the target order improved test accuracy and the proposed method clearly discovered a beneficial curriculum.

For the time-series data sets the leave-one-out testing regime did not involve varying the sample size so we have instead presented the results in Tables 2 and 3. Again we see that imposing a target curriculum improved the generalisation performance of some targets later in the suspected order even for shallow hierarchies and with extremely limited sample sizes. In Table 3 we see an improvement in the target G6 and in Table 2 we see improvement for both subsequent targets—Cdc2/Cdc13 and Cdc2/Cdc13*—in one of the two estimated hierarchies (see Figure 2).
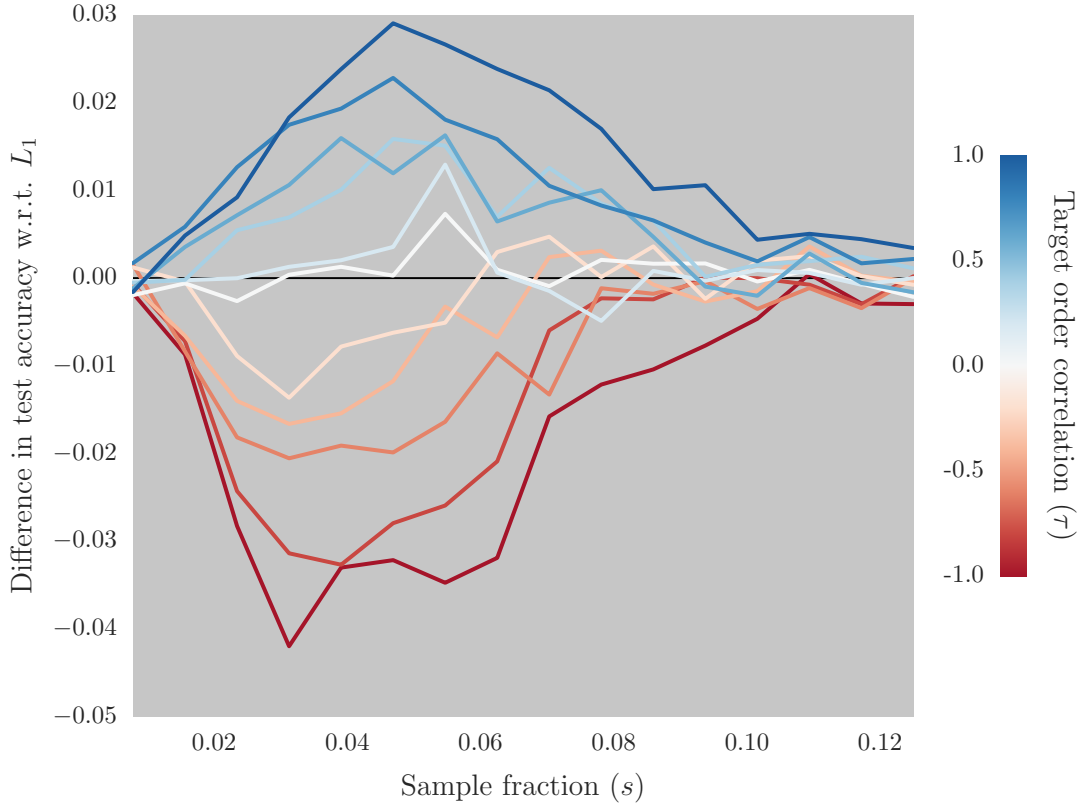
Figure 6: Mean difference in test accuracy between L_{gh} and L_1 over various training set sizes for the 5-bit addition problem for randomised target orderings. Each line represents the mean change in test accuracy for each possible level of correlation between the randomly given order and the known order. For an n-element permutation there are $\lceil n \times (n-1)/2 \rceil$ possible values of τ, hence 11 lines. The central permutations offer little benefit, and since the random permutations are skewed heavily toward $\tau = 0$ we can infer that a randomly generated ordering would offer no improvement on average. We see very clearly that that direct opposite $(\tau = -1)$ of the suggested curricula impeded the learning performance by as much as the correct order improved it and that the exact easy-to-hard ordering $(\tau = 1)$ conferred the largest benefit.
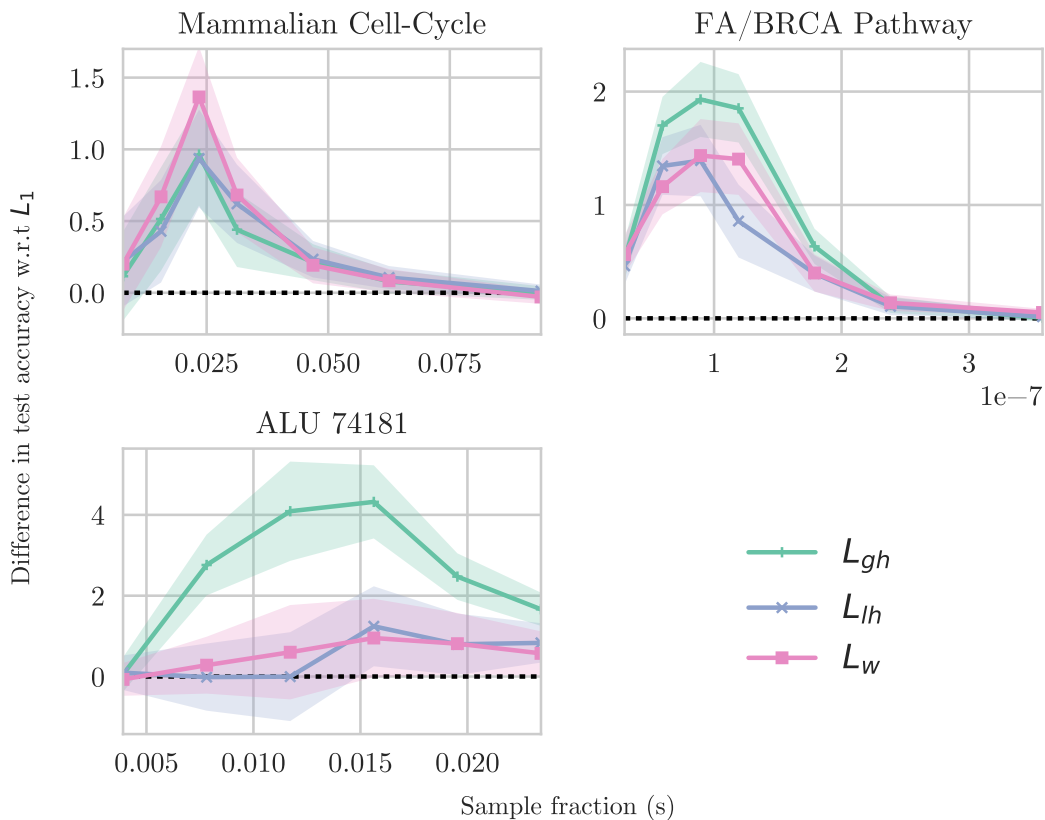
Figure 7: Mean difference in test accuracy between L_1 (dotted baseline) and each of L_w, L_{lh} and L_{gh} on each training sample for the 74181, mammalian cell-cycle and FA/BRCA pathway models. The 95% confidence interval of the mean is shown with transparent bands. On all problems we see the same net increase in test accuracy—again varying widely between problems. We see again that, except in the case of the mammalian cell-cycle model for which the losses are not clearly distinguishable, this improvement increases as the loss more strictly enforces the ordering ($L_w \to L_{lh} \to L_{gh}$).

	SK	Ste9	Rum1	PP	Cdc2/Cdc13	Cdc2/Cdc13*
L_w	0.4	0.5	0.4	1.6	1.2	0.6
L_{lh}	0.6	0.5	0.4	0.8	1.0	0.1
L_{gh}	0.9	0.5	0.3	0.5	1.3	1.4

Table 2: Results for the yeast data set. See Figure 2 for the estimated hierarchies. The SK→ {Ste9, Rum1} hierarchy has given little improvement, however we see more promise in the PP→ {Cdc2/Cdc13, Cdc2/Cdc13*} hierarchy, particularly for L_{gh}.

	G2	G8	G6
L_w	0.1	0.2	0.8
L_{lh}	0.7	-0.9	1.3
L_{gh}	0.1	0.1	1.4

Table 3: Results for the *E. coli* data set. See Figure 2 for the estimated hierarchies. The {G2, G8}→G6 hierarchy has given some improvement, however for L_{lh} we see a drop in performance on one of the base targets in the hierarchy. For L_{gh} the results are still overall positive.

4. Discussion and Future Work

We have shown that loss functions that impose a target order improve the overall generalisation performance of FBNs using a series of hierarchical multi-target circuit-inference problems. We also showed this improvement increases as the target order is more strictly enforced. Note that the losses considered do not alter the set of global optima. This approach does not restrict the space of networks but instead modifies their probability of discovery in a natural way using concepts inspired by human teaching and learning practices.

The set of circuit-inference problems described in Section 3.1.1 all possess a common element: intermediate values which are useful for computing the $m + 1^{th}$ output are computed for the mth output. A human designer with access to a complete m-bit circuit would have a much easier time building an $(m + 1)$-bit circuit. It is natural to assume that this ease should extend—in terms of convergence speed, generalisation or both—to a optimiser which builds such circuits in the correct order. We observed the latter but not the former.

We observed that performance improvements primarily occur around the existent generalisation phase transition. This is intuitive: there are fundamental limits with respect to training-set size that cannot be alleviated, and there are also sizes for which there is enough information present that teaching a target order confers no advantage. What we have shown is that—in the interim region between too little and plenty of training information— teaching targets in order of difficulty can make better use of what data is present. This improvement is expected in light of the positive effects of hierarchical cost functions on optimiser performance (Moscato, 1993).

The improvement on the final target from L_{gh} actually decreases (bottom of Figure 3). Initially, this appears as a cause for concern as it seems to indicate a point of diminishing returns. However this happened consistently only on the final target for all problems. Were

there a per-problem tipping point we would expect to see it before the last target, or not at all, on some problems. This is also the same for varying sizes of binary addition (see Online Appendix 2).

A more likely explanation is the gate budget coupled with a lack of constraints on which gates each target can source from. The loss function for which we see the issue occurring, L_{gh} essentially forces optimisation of each target in succession. As the process continues, it becomes increasingly more likely that any given target will have a majority of its computational nodes closer to the end of the ordering of gates. Subsequent sub-networks will thus require later and later nodes to be available in order to correctly discover the hierarchical structure of the problem which becomes an issue toward the end when the gate budget is exhausted. This could be alleviated by reserving nodes for each target.

We have also suggested an intuitive approach to determining the target overlap for Boolean MLCs, and to automatically determine an appropriate target order. For the problems considered, the bulk of the generalisation improvement conferred by the curriculum remained, even when no information on the true target order was provided to the learner.

We further examined the same loss functions in conjunction with the target curricula estimation method on three models of real world phenomena as well as on two regulatory network time-series data sets. On all three models we observed test accuracy increases similar to that seen on the circuit-inference problems. In the case of the yeast and *E. coli* data set we discovered possible target hierarchies which were shallow and as such we expected less improvement (Figure 3 shows that generalisation gains are focused primarily on later targets). Nonetheless the results are still in line with what we have observed for the previous problems. This is promising as we have not explicitly accounted for noise in either the curricula detection method or design of the loss functions.

Finally we examine the importance of the curricula itself in Section 3.4.2. We see that the easy-to-hard curricula gave the strongest improvement, that curricula which were anti-correlated with the easy-to-hard permutation actually decreased test accuracy, and that the curricula which were uncorrelated (i.e. essentially random) gave no improvement over learning with no curriculum. The fact that the peak in the distribution of permutations over τ occurs at zero means that randomly generated curricula will most commonly give no benefit and thus highlights the importance of principled methods for determining target curricula.

Not discussed here is the effect that the use of these guiding functions has on the average convergence time. We observed a variable effect on training speed and, for some problems, there is a cost in convergence speed when using target curricula. More information on training time is available in Online Appendix 2.

Another possible impact on training speed is the requirement for the target ordering method to solve an NP-hard optimisation problem. However, for all problems considered, the target order detection process had almost no impact on overall training time. For significantly larger problem instances, however, heuristic solvers would likely be required.

Noise has not been considered as a factor in this work. It is uncertain whether the minFS based methods for estimating target overlap and target schedules—or FBNs themselves—are robust to incorrect target labels. Future work should address this shortfall, perhaps by allowing the per-target error constraints in L_{gh} to be non-zero. The drop in performance on

the final target should also be addressed; possibly by using stratified networks, or annealed depth constraints, to prevent later gates being consumed by earlier targets.

While our results are only for the case of Boolean MLCs learned with FBNs, we expect they may extend to other non-convex learners, such as deep neural nets, for problems with categorical or Boolean domains, provided differentiable analogues to L_{lh} and L_{gh} are found. Future work could also examine target feature selection methods for real valued domains, as well as heuristic Boolean feature selection methods, to determine if the approach presented here is more generally applicable. A recent discussion of the use of intrinsic dimension as a proxy for the overall complexity of arbitrary data sets (Granata and Carnevale, 2016) is promising.

Acknowledgments

References

Yaser S Abu-Mostafa. Learning from hints in neural networks. *Journal of Complexity*, 6 (2):192–198, June 1990.

Shohag Barman and Yung-Keun Kwon. A novel mutual information-based Boolean network inference method from time-series gene expression data. *PLOS ONE*, 12(2):e0171097, February 2017.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *International Conference on Machine Learning (ICML)*, pages 41–48. ACM, 2009.

Edmund K. Burke and Yuri Bykov. A late acceptance strategy in hill-climbing for examination timetabling problems. In *International Conference on the Practice and Theory of Automated Timetabling (PATAT)*, 2008.

Rich Caruana. Multitask learning. *Machine Learning*, 28(1):41–75, 1997.

Josu Ceberio, Alexander Mendiburu, and Jose A. Lozano. The linear ordering problem revisited. *European Journal of Operational Research*, 241(3):686–696, March 2015.

Carlos Cotta and Pablo Moscato. The k-feature set problem is W[2]-complete. *Journal of Computer and System Sciences*, 67(4):686–690, December 2003.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or -1. *arXiv:1602.02830 [cs]*, February 2016.

Scott Davies and Stuart Russell. NP-Completeness of searches for smallest possible feature sets. In *AAAI Symposium on Intelligent Relevance*, pages 37–39. AAAI Press, 1994.

Alireza Goudarzi, Christof Teuscher, Natali Gulbahce, and Thimo Rohlf. Learning, generalisation, and functional entropy in random automata networks. *International Journal of Autonomous and Adaptive Communications Systems*, 7(3):295–314, 2014.

Daniele Granata and Vincenzo Carnevale. Accurate estimation of the intrinsic dimension using graph distances: Unraveling the geometric complexity of datasets. *Scientific Reports*, 6:31377, August 2016.

Çağlar Gülçehre and Yoshua Bengio. Knowledge matters: Importance of prior information for optimization. *Journal of Machine Learning Research*, 17(8):1–32, 2016.

Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander G. Hauptmann. Self-paced curriculum learning. In *AAAI Conference on Artificial Intelligence*, pages 2694–2700, Austin, Texas, 2015. AAAI Press.

Faisal Khan, Bilge Mutlu, and Xiaojin Zhu. How do humans teach: On curriculum learning and teaching dimension. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1449–1457. Curran Associates, Inc., 2011.

M. Pawan Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In *International Conference on Neural Information Processing Systems*, NIPS'10, pages 1189–1197, USA, 2010. Curran Associates Inc.

Abhimanyu Lad, Rayid Ghani, Yiming Yang, and Bryan Kisiel. Toward optimal ordering of prediction tasks. In *SIAM International Conference on Data Mining*, Proceedings, pages 884–893. SIAM, April 2009.

Changsheng Li, Junchi Yan, Fan Wei, Weishan Dong, Qingshan Liu, and Hongyuan Zha. Self-paced multi-task learning. In *AAAI Conference on Artificial Intelligence*, pages 2175–2181, 2017.

Pablo Moscato. An introduction to population approaches for optimization and hierarchical objective functions: A discussion on the role of tabu search. *Annals of Operations Research*, 41(2):85–121, June 1993.

Stefano Patarnello and Paolo Carnevali. Learning networks of neurons with Boolean logic. *Europhysics Letters*, 4(4):503, 1987.

Anastasia Pentina, Viktoriia Sharmanska, and Christoph H. Lampert. Curriculum learning of multiple tasks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5492–5500, June 2015.

Arnaud Poret and Jean-Pierre Boissel. An in silico target identification using Boolean network attractors: Avoiding pathological phenotypes. *Comptes Rendus Biologies*, 337 (12):661–678, December 2014.

Jesse Read, Bernhard Pfahringer, Geoff Holmes, and Eibe Frank. Classifier chains for multi-label classification. *Machine Learning*, 85(3):333–359, June 2011.

Mateus Rocha de Paula, Regina Berretta, and Pablo Moscato. A fast meta-heuristic approach for the α-β-k-feature set problem. *Journal of Heuristics*, 22(2):199–220, December 2015.

Valentin I. Spitkovsky, Hiyan Alshawi, and Daniel Jurafsky. Baby steps: How "less is more" in unsupervised dependency parsing. In *NIPS: Grammar Induction, Representation of Language and Language Learning*, 2009.

Christof Teuscher, Natali Gulbahce, and Thimo Rohlf. Learning and generalization in random Boolean networks. In *Dynamic Days 2007: International Conference on Chaos and Nonlinear Dynamics*, Boston, MA, January 2007.

Alan M. Turing. *Intelligent Machinery. Report for National Physical Laboratory. Reprinted in Ince, Dc (Editor). 1992. Mechanical Intelligence: Collected Works of A.M. Turing.* Amsterdam: North Holland, 1948.

Christian Van den Broeck and Ryoichi Kawai. Learning in feedforward Boolean networks. *Physical Review A*, 42(10):6210, 1990.

Xin Wang, Jinbo Bi, Shipeng Yu, Jiangwen Sun, and Minghu Song. Multiplicative multitask feature learning. *Journal of Machine Learning Research*, 17(80):1–33, 2016.