Learning Graph Weighted Models on Pictures

Philip Amortila   PHILIP.AMORTILA@MAIL.MCGILL.CA
Guillaume Rabusseau   GUILLAUME.RABUSSEAU@MAIL.MCGILL.CA
Reasoning and Learning Lab - School of Computer Science - McGill University

Editors: Olgierd Unold, Witold Dyrka, and Wojciech Wieczorek

Abstract

Graph Weighted Models (GWMs) have recently been proposed as a natural generalization of weighted automata over strings and trees to arbitrary families of labeled graphs (and hypergraphs). A GWM generically associates a labeled graph with a tensor network and computes a value by successive contractions directed by its edges. In this paper, we consider the problem of learning GWMs defined over the graph family of pictures (or 2-dimensional words). As a proof of concept, we consider regression and classification tasks over the simple *Bars & Stripes* and *Shifting Bits* picture languages and provide an experimental study investigating whether these languages can be learned in the form of a GWM from positive and negative examples using gradient-based methods. Our results suggest that this is indeed possible and that investigating the use of gradient-based methods to learn picture series and functions computed by GWMs over other families of graphs could be a fruitful direction.

Keywords: Graph Weighted Models, Picture Series, Learning Weighted Automata, Structured Data

1. Introduction

The heart of automata theory is the modeling and study of functions defined over syntactical structures such as strings, trees, or graphs. A classic example are Weighted Finite Automata (WFA) (Schützenberger, 1961; Berstel and Reutenauer, 1988), which are finite state machines computing real-valued functions over strings. WFA encompass a wide class of useful tools for predictions such as hidden Markov models, predictive state representations, and probabilistic automata and are thus particularly relevant to the machine learning community. Of great interest, specifically for machine learning, are the so-called spectral learning algorithms (Bailly et al., 2009; Hsu et al., 2009; Balle et al., 2014), which are efficient and consistent algorithms for learning WFAs.

Various extensions of WFA have been proposed such as Weighted Tree Automata (WTA) (Berstel and Reutenauer, 1982) and Weighted Picture Automata (WPA) (Bozapalidis and Grammatikopoulou, 2005), which model functions defined over trees and pictures respectively. Graph Weighted Models (GWM), introduced in (Bailly et al., 2015, 2017), are a natural generalization to the case of labelled graph inputs, and include the above models as special cases. Roughly speaking, a GWM is defined by associating a tensor with every character of a ranked alphabet and computes a real-valued function over graphs labelled by symbols of the alphabet. The value computed by a GWM is obtained by construct-

ing a *tensor network* out of the graph input and computing a value via successive tensor contractions.

To date, GWMs have been studied mostly from the perspective of formal languages (Bailly et al., 2017; Rabusseau, 2018), and designing learning algorithms for these automata remains to be done. In particular, extending the spectral algorithms for WFA to GWMs remains an open (and challenging) problem which we are currently investigating. In this work, we examine an alternative approach for learning GWMs from data by experimentally exploring how traditional gradient-based algorithms perform on this task. While this approach can be applied to GWMs defined over arbitrary families of graphs, we focus here on regression and classification problems for simple picture languages. Firstly, as an instance of regression tasks, we examine the *Bars & Stripes* language made of pictures containing only horizontal or vertical stripes. After showing that this language can be computed by a WPA (and equivalently by a GWM defined over pictures), we show empirically that minimizing the mean squared error over a large enough sample of input/output examples allows one to recover a GWM approximating the function of interest which generalizes to unseen pictures of different sizes. Secondly, akin to logistic regression, we show how to handle classification tasks by composing the output of the model with a sigmoid activation function, and empirically demonstrate that minimizing the cross-entropy error can lead to successful classification of the *Shifting Bits* language, which consists of pictures where each row is a horizontal shift of the previous one.

2. Preliminaries

Firstly, we introduce the necessary notions. In this paper we are working with picture data, thus we begin by discussing Weighted Picture Automata.

2.1. Weighted Picture Automaton

A *picture* (also called an *image* or a *2d-word*) $p \in \mathcal{P}$ over a finite alphabet Σ is defined as a non-empty rectangular array of elements of Σ, formally $\mathcal{P} = \cup_{m,n \geq 1}\Sigma^{m \times n}$. Given $p \in \Sigma^{m \times n}$, we write $p_{i,j}$ for the component of p at position (i,j). A *picture language* is a set of pictures, while a *picture series* is a function from $\mathcal{P}$ to a commutative semiring. Regular picture languages can equivalently be described in terms of automata, sets of tiles, rational operations, or monadic second order logic (Giammarresi and Restivo, 1996; Giammarresi et al., 1996; Inoue and Nakamura, 1977; Latteux and Simplot, 1997). The extension of regular picture languages to the quantitative setting led to the definition of *recognizable picture series* whose theoretical study has been of recent interest (Bozapalidis and Grammatikopoulou, 2005; Mäurer, 2005; Fichtner, 2011; Babari and Droste, 2015). Recognizable picture series were first introduced in (Bozapalidis and Grammatikopoulou, 2005) by means of Weighted Picture Automata.

Definition 1 (Bozapalidis and Grammatikopoulou (2005)) *A Weighted Picture Automaton (WPA) over an alphabet Σ is a tuple $A = (Q, R, F_w, F_n, F_e, F_s, \delta)$, where Q is a finite set of states, $R \subseteq \Sigma \times Q^4$ is a finite set of rules, $F_w, F_n, F_e, F_s \subseteq Q$ are four*

poles of acceptance, and $\delta : R \to \mathbb{R}$ is the weighted transition function[1]. Given a rule $r = (\sigma, q_w, q_n, q_e, q_s) \in R$, we denote its label by $\ell(r) = \sigma$, and its poles by $west(r) = q_w$, $north(r) = q_n$, $east(r) = q_e$, and $south(r) = q_s$. Given an image $p \in \Sigma^{m \times n}$, a run $c \in R^{m \times n}$ is an assignment of rules such that the following compatibility properties hold:

$$south(c_{i,j}) = north(c_{i+1,j}) \ \forall i \in [m-1], \forall j \in [n]$$
$$east(c_{i,j}) = west(c_{i,j+1}) \ \forall i \in [m], \forall j \in [n-1]$$

and $\ell(c_{i,j}) = p_{i,j} \ \forall i \in [m], j \in [n]$, where $[n] := \{1, ..., n\}$. A run is accepted if the outer poles are in their respective poles of acceptance, i.e.

$$west(c_{i,1}) \in F_w, \ south(c_{m,j}) \in F_s, \ east(c_{i,n}) \in F_e, \ north(c_{1,j}) \in F_n \ \forall i \in [m], \forall j \in [n]$$

We denote by $R(p)$ the set of accepted runs on p. The weight function is extended to runs via $\delta(c) = \Pi_{i,j} \delta(c_{i,j})$. The function computed by A is the sum of the weights over all accepting runs: $f_A(p) = \sum_{c \in R(p)} \delta(c)$, with the convention that $f_A(p) = 0$ if there are no accepting runs.

2.2. Tensor Networks and Graph Weighted Models

We first briefly introduce *tensors* and *tensor networks*. For our purposes, a tensor over $\mathbb{R}$ of order p can be seen as a p-dimensional array of scalars: $\left(\mathcal{T}_{i_1, i_2, ..., i_p} \in \mathbb{R} : i_n \in [d], n \in [p] \right)$[2]. Tensor networks are a useful tool for visualizing operations on tensors and will simplify our exposition of GWMs. A tensor network is an undirected graph where the nodes correspond to tensors and the outgoing edges correspond to the different dimensions of the tensor. For example, a vector is a node of degree 1 and a matrix is node of degree 2. We allow nodes to have free edges that do not connect to other nodes. Each edge is numbered by an index corresponding to a dimension of the tensor, such a numbered edge is called a *port*. Connecting two ports in a tensor network corresponds to a summation of the two tensors along the connected indices. A few examples of tensor networks and their associated computations are shown in Figure 1.

Formally, one way to compute the tensor represented by a tensor network by first taking the tensor product of the tensors associated with all the nodes in the graph, and then performing the contractions associated to the edges, i.e. summing over the pairs of indices corresponding to connected ports. For example, for the tensor network in Figure 1 (middle), the tensor product of $\mathbf{A}$ and $\mathbf{B}$ is given by the fourth order tensor $(\mathbf{A} \otimes \mathbf{B})_{i_1 i_2 i_3 i_4} = \mathbf{A}_{i_1 i_2} \mathbf{B}_{i_3 i_4}$, and one then performs the contraction between the second port of $\mathbf{A}$ (i_2) and the first port of $\mathbf{B}$ (i_3) to obtain the second order tensor $\sum_j \mathbf{A}_{i_1 j} \mathbf{B}_{j i_4}$. Note first that in a graph with more than one edge the order of contractions does not matter; second, this is a naive way of performing the computation: one does not need to construct the whole tensor product before performing the contractions, however finding the optimal contraction sequence is an NP-hard problem (Pfeifer et al., 2014).

---

1. WPAs are originally defined over arbitrary commutative semi-rings but we will only consider the field $\mathbb{R}$.

2. In general we could have $i_n \in [d_n]$ for any list of integers $d_1, d_2, ..., d_p$, however here we only consider the case $d_n = d \ \forall n$ – these are called *hypercubic* tensors.
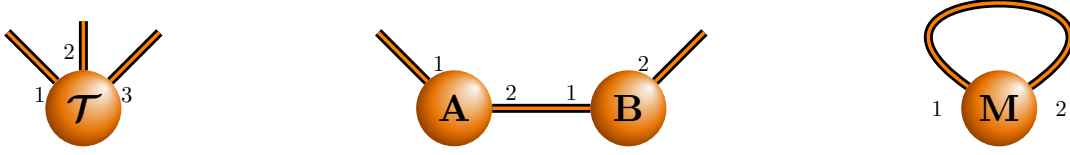
Figure 1: (left) The arity of a node is the order of the tensor, here the node denotes a 3rd-order tensor $\mathcal{T} \in \mathbb{R}^{d \times d \times d}$. (middle) The connection of two ports corresponds to a contraction over the corresponding indices, here connecting the second port of the matrix $\mathbf{A}$ with the first port of the matrix $\mathbf{B}$ represents the classical matrix product: $(\mathbf{AB})_{i_1,i_2} = \sum_k \mathbf{A}_{i_1,k} \mathbf{B}_{k,i_2}$. (right) Similarly, connecting the two ports of a matrix represents the trace operation: $\mathrm{Tr}(\mathbf{M}) = \sum_k \mathbf{M}_{kk}$.

Graph Weighted Models have been introduced as computational models over labelled graphs in (Bailly et al., 2015, 2017). A Graph Weighted Model associates a labelled graph with a tensor network computing the corresponding value. Here we present only the specifics necessary for computations of GWMs over pictures.

Definition 2 (Graph Weighted Models) *A GWM M (over $\mathbb{R}$) on pictures is a tuple $M = (d, \{\mathcal{T}^\sigma\}_{\sigma \in \Sigma}, \alpha^w, \alpha^n, \alpha^e, \alpha^s)$, where d is the dimension (or number of states), $\mathcal{T}^\sigma \in \mathbb{R}^{d \times d \times d \times d}$ is a tensor of order 4 for each $\sigma \in \Sigma$, and $\alpha^w, \alpha^n, \alpha^e, \alpha^s \in \mathbb{R}^d$ are the border vectors. Given a picture p, the function computed by M is the tensor network obtained from p by replacing every character with its associated tensor and adding the border vectors.*

As an example, the value computed by a GWM M on the picture $p = \begin{smallmatrix} a & b \\ b & a \end{smallmatrix}$ is represented as a tensor network in Figure 2.
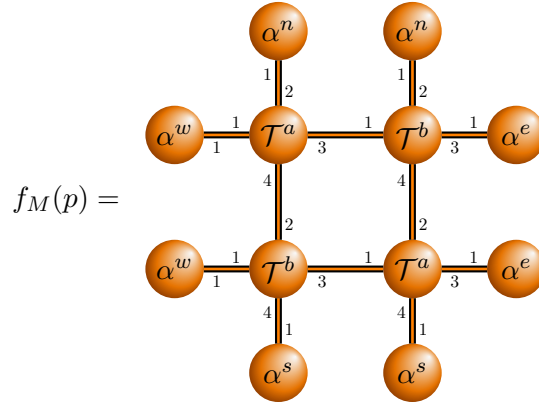

Figure 2: The tensor network associated with the picture $p = \begin{smallmatrix} a & b \\ b & a \end{smallmatrix}$. Note that the tensor network has no free edges, thus indeed computes to a real number.

The computation can be written explicitly as:

$$f_M(p) = \sum_{i_1,i_2,..,i_{12}} \alpha_{i_1}^w \alpha_{i_2}^n \mathcal{T}_{i_1,i_2,i_3,i_4}^a \mathcal{T}_{i_3,i_5,i_6,i_7}^b \alpha_{i_5}^n \alpha_{i_6}^e \mathcal{T}_{i_8,i_4,i_9,i_{10}}^b \alpha_{i_8}^w \alpha_{i_{10}}^s \mathcal{T}_{i_9,i_7,i_{11},i_{12}}^a \alpha_{i_{11}}^e \alpha_{i_{12}}^s \quad (1)$$

The following useful result states that any WPA can be realized by a GWM on pictures.

Proposition 3 *(Rabusseau, 2016, Proposition 4) Any function that is computable by a WPA with d states is computable by a d-dimensional GWM on pictures.*

We note that the converse of the previous proposition holds in the sense that WPA can compute any function that can be computed by a GWM defined on the family of graph representations of pictures (see (Bailly et al., 2017, Proposition 3.9) for more details).

3. Learning of GWMs on Pictures

In this section we present the languages we will attempt to learn in our experiments. Afterwards, we formalize the learning problem, and present the gradient–based approach we propose to tackle it.

3.1. Bars & Stripes and Shifting Bits

We will apply our methods to regression and classification tasks on two picture languages over 2-letter alphabets (denoted a for white and b for black), demonstrating the ability of our approach to learn simple picture languages.

3.1.1. BARS & STRIPES

In the *Bars & Stripes* (BS) language (MacKay, 2003), each image is composed of either horizontal stripes or vertical bars, but not both (unless it is fully white or fully black). Formally, we have $BS = \{p \in \mathcal{P} : (p_{i,j} = p_{i+1,j} \ \forall i,j) \text{ or } (p_{i,j} = p_{i,j+1} \ \forall i,j)\} \subseteq \mathcal{P}$. A few sample images are shown in Figure 3.
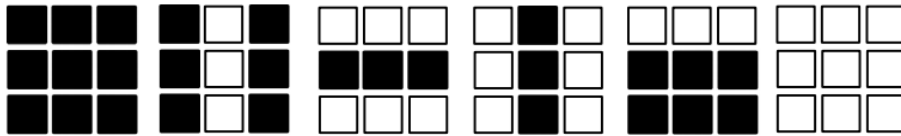

Figure 3: Examples of 3×3 images in the Bars & Stripes language. This figure and the next from (Melchior et al., 2016)

3.1.2. SHIFTING BITS

In the *Shifting Bits* (SB) language, inspired from (Melchior et al., 2016), each row in the pictures is a horizontal translation of the previous row. We allow for shifts of arbitrary length. Formally, we have $SB = \cup_{m,n} SB_{m \times n}$, where $SB_{m \times n} = \{p \in \Sigma^{m \times n} : \exists s \in [m]$ such that $(p_{i+1,j} = p_{i,j-s}$ if $j - s \geq 1$ else $p_{i+1,j} = b)\}$. Some examples of shift images are given in figure 4.
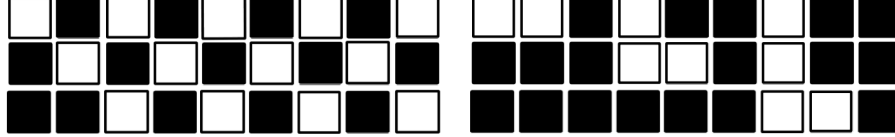

Figure 4: Examples of 3×9 images in the Shifting Bits language, with shifts of size 1 and 3.

3.2. Recognizability

Before considering the problem of learning these languages from positive and negative examples, we first show that the Bars & Stripes language is indeed recognizable by a WPA.

Proposition 4 (BS is WPA-recognizable) *The Bars & Stripes language $BS \subseteq \mathcal{P}$ is WPA-recognizable, in the sense that there exists a WPA whose support is the Bars & Stripes language.*

More precisely, there exists a WPA A with 6 states such that

$$f_A(p) = \begin{cases} 1 & \text{if } p \in \mathcal{P} \text{ and } p \text{ contains at least one a and one b} \\ 2 & \text{if } p \in \mathcal{P} \text{ and } p \text{ contains only a's or only b's} \\ 0 & \text{otherwise.} \end{cases}$$

Proof Consider the WPA $A = (Q, R, F_w, F_e, F_s, F_n, \delta)$, where $Q = \{q_0, q_{a\rightarrow}, q_{a\downarrow}, q_{b\rightarrow}, q_{b\downarrow}, q_f\}$, $F_w = F_n = q_0$, and $F_e = F_s = q_f$. Intuitively, q_0 and q_f are the initial and final states, the horizontal states $q_{a\rightarrow}, q_{b\rightarrow}$ encode whether we are in a picture with horizontal bars, and similarly for the vertical states $q_{a\downarrow}, q_{b\downarrow}$. For succinctness, we represent the rule set R via the following diagrams:
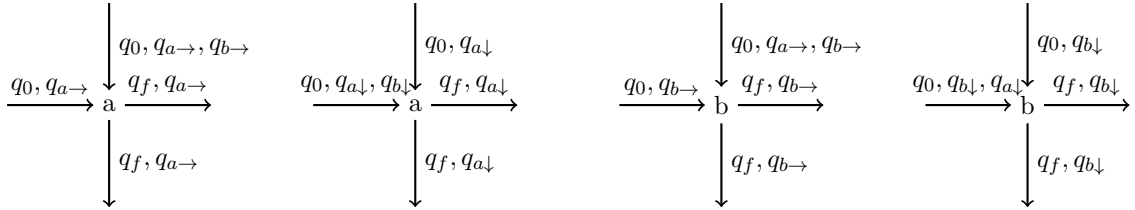
The diagrams indicate that when the label in the center is observed, any combination of the above states are valid for the poles of the rule. The first picture, for example, says that

when the white label is observed, a valid run has poles $west(a) \in \{q_0, q_{a\rightarrow}\}$, $north(a) \in \{q_0, q_{a\rightarrow}, q_{b\rightarrow}\}$, $east(a) \in \{q_f, q_{a\rightarrow}\}$, and $south(a) \in \{q_f, q_{a\rightarrow}\}$. Finally, the transition function is simply $\delta(r) = 1 \ \forall r \in R$. Thus, $\delta(c) = \Pi_{i,j}\delta(c_{i,j}) = 1$ for an accepting run and $f_A(p)$ simply counts the number of accepting runs on an image p.

First we show that if p is an image in BS, then there exists an accepting run for p. Assume $p \in \{a, b\}^{m \times n}$ is horizontally striped, and let σ_i be the common label of the ith row. Consider the run c defined by

$$north(c_{1,j}) = west(c_{i,1}) = q_0 \ \forall i \in [m], j \in [n]$$
$$east(c_{i,n}) = south(c_{m,j}) = q_f \ \forall i \in [m], j \in [n]$$
$$east(c_{i,j}) = south(c_{i,j}) = q_{\sigma_i\rightarrow} \ \forall(i,j) \in ([m] \times [n]) \setminus \{(m,n)\}$$

To ensure the compatibility properties of the run c we also set

$$west(c_{i,j}) = east(c_{i,j-1}) = q_{\sigma_i\rightarrow} \ \forall \ 2 \leq j \leq n, i \in [m]$$
$$north(c_{i,j}) = south(c_{i-1,j}) = q_{\sigma_{i-1}\rightarrow} \ \forall \ 2 \leq i \leq m, j \in [n]$$

Now, if p is not fully white or fully black, then c is the only possible accepting run on this image, since every non-border rule $c_{i,j}$ can only output $east(c_{i,j}) = south(c_{i,j}) = q_{p_{i,j}\rightarrow}$. Thus $f_A(p) = 1$. The proof is analogous if p is vertically striped. Lastly, in the case where the image is fully white (resp. fully black), we have $f_A(p) = 2$, since an accepting run can either assign $q_{a\downarrow}$ or $q_{a\rightarrow}$ (resp. $q_{b\downarrow}$ or $q_{b\rightarrow}$) to every pole of the non-border labels.

Now we show that if the automaton A accepts p, then $p \in BS$. To show the contrapositive, assume $p \notin BS$. Then, somewhere in p there must exist a 2×2 square of labels of the form $\begin{smallmatrix} \star & a \\ a & b \end{smallmatrix}$ where $\star \in \{a, b\}$ (up to rotation and/or bit flip of the symbols). Since (i) there are no rules allowing both the north and west poles of b to simultaneously have poles $q_{a\downarrow}$ or $q_{a\rightarrow}$ and (ii) the south and east poles of the a's are necessarily in one of these states, there does not exist an accepting run for p and $f_A(p) = 0$. This concludes the proof. ∎

It follows from Propositions 3 and 4 that there exists a GWM recognizing the BS language. In Section 3.3 we show how to learn a GWM computing this target function via empirical risk minimization. However, we observe that the particular GWM representation associated to the WPA given in the previous proposition is not unique, by a reasoning similar to the proof of (Rabusseau, 2016, Proposition 10): loosely speaking, the value corresponding to a tensor network associated with a given picture is invariant under a change of basis of every tensor. More specifically, for any invertible matrix $\mathbf{P}$, we can consider the transformation of the tensors $\boldsymbol{\mathcal{T}}^\sigma$ described in Figure 5, with analogous transformations for the border vectors $\boldsymbol{\alpha}^\tau$ for $\tau \in \{n, e, s, w\}$. One can verify that applying this transformation to all the tensors in a GWM gives a new model which computes the same value on all pictures since all $\mathbf{P}$ matrices will contract with their inverses.

We note that for fixed m, n the Shifting Bits language $SB_{m \times n}$ is also recognizable – intuitively speaking we can use n states to count that the shift property is being preserved in each row with a construction similar to the previous proposition. By (Rabusseau, 2016, Proposition 5), finite unions $\cup_{i=1}^k S_{m_i \times n_i}$ are also recognizable. However, we will not be leveraging a recognizability result for this language, since rather than trying to recover an automaton that recognizes the language, we will use this language as an example to
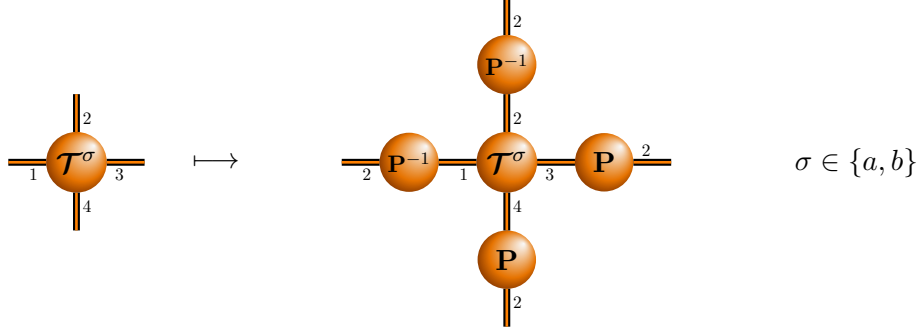
Figure 5: Transforming all tensors $\mathcal{T}^\sigma$ through this map, with an analogous map for the border tensors $\boldsymbol{\alpha}^\tau$, gives a GWM which computes the same value for every picture.

show how GWMs can be used to model conditional probabilities via methods described in Section 3.3.

3.3. The Learning Problem

We consider the problem of approximating a target function from a finite number N of input/output examples $S = \{(p_i, f(p_i))\}_{i=1}^N \subset \mathcal{P} \times \mathbb{R}$, where f is the target function of interest. To find a candidate GWM $\hat{M}$ which approximates the desired function, we optimize an error function over the training samples with gradient descent. We now outline the specifics for the regression and classification tasks, respectively.

For the Bars & Stripes language we seek $f = f_M$ where f_M is the function computed by the GWM recognizing BS from Proposition 4, i.e. the function which outputs 0 on pictures not in BS, 1 on pictures which have only horizontal or vertical stripes, and 2 on pictures which are fully white or fully black. In this setup the candidate GWM $\hat{M}$ is such that $f_{\hat{M}} \simeq f_M$. We minimize the mean squared error (MSE) over the training data.

$$J_{\text{MSE}}(\hat{M}, S) = \frac{1}{|S|} \sum_{(p, f_M(p)) \in S} (f_M(p) - f_{\hat{M}}(p))^2$$

For the Shifting Bits language, we consider the binary classification task which is to predict 1 on pictures in SB and 0 otherwise. As with other commonly used classification models, such as logistic regression and neural networks (Bishop, 2006), we use a sigmoid function $\sigma : x \mapsto 1/(1 + e^{-x})$ at the output of the GWM in order to model the conditional probability that the picture p is in SB: $\mathbb{P}(f(p_i) = 1 | p_i) \simeq \sigma(f_{\hat{M}}(p_i))$. The prediction made in the classification setting is the class with the highest probability, obtained by rounding the output of the sigmoid. The target function is thus $f = \mathbb{1}_{[SB]}$, the indicator function for the Shifting Bits set. To learn the conditional probability, we use maximum likelihood estimation, which is equivalent to minimizing the cross entropy (CE) between the target

function and the candidate

$$J_{\text{CE}}(\hat{M}, S) = -\frac{1}{|S|} \sum_{(p, f(p)) \in S} f(p) \log \sigma \left(f_{\hat{M}}(p) \right) + (1 - f(p)) \log \left(1 - \sigma \left(f_{\hat{M}}(p) \right) \right). \quad (2)$$

The optimization of both error functions is done using a stochastic gradient descent approach. More precisely, at each iteration t, a random mini-batch S_t is uniformly drawn from S and each parameter of $\hat{M} = (d, \mathcal{T}^a, \mathcal{T}^b, \alpha^w, \alpha^e, \alpha^s, \alpha^n)$ is updated by taking a small step in the opposite direction of the gradient. We use the Adam optimizer (Kingma and Ba, 2014) to update the parameters. It is worth mentioning that while deriving an analytic expression for the gradient $\nabla_{(\mathcal{T}^a, \mathcal{T}^b, \alpha^w, \alpha^e, \alpha^s, \alpha^n)} J(\hat{M}, S_t)$ may be a tedious task, modern deep learning frameworks such as PyTorch (Paszke et al., 2017) can make the implementation of the minimization algorithm relatively uncomplicated by using automatic differentiation techniques to numerically evaluate the gradient via backpropagation. We remark that the optimization problem is highly non-convex with respect to the weights of the model due to the numerous multiplicative interactions involved in the computations made by the GWM (recall equation 1). Moreover, by the non-uniqueness remarks of Section 3.1, we can perform a change of basis in the style of Figure 5 to obtain an equivalent model with the same error. This implies that there exist infinitely many global minima to the objective functions, and that if there exists at least one local minimum, then there also exist infinitely many. Thus, convergence to a global minimum is not guaranteed when using gradient descent methods to solve these optimization problems.

4. Experiments

We implemented the learning procedures detailed in Section 3.3, and performed experiments on training samples generated from the Bars & Stripes and the Shifting Bits languages. Our experiments for both languages consider the effect of varying training set sizes and the ability of the models to generalize to unseen pictures of different image dimensions. Our findings are the following: given a sufficient number of training pictures, the learned GWMs are able to (i) make accurate predictions on unseen pictures of the same size and (ii) generalize and make accurate predictions on higher image sizes. First, we remark that the datasets are highly class imbalanced: for pictures of size $m \times n$ there are only $2^m + 2^n - 2$ and $n2^n - 1$ samples (out of 2^{mn}) which are positive in the Bars & Stripes and Shifting Bits languages, respectively. To ensure that the model does not learn the constant function $f = 0$, we include a 50% split of positive and negative pictures in the training and test sets. The training and testing sets are kept disjoint whenever possible (i.e. whenever there are enough positive samples to allow for no overlap). In all experiments we use the Adam optimizer. We begin by detailing the setup and the experimental results for the Bars & Stripes language.

4.1. Bars & Stripes

The training and test sets are drawn from the same distribution, however we ensure that the negative samples of the two sets are disjoint so as to accurately test the generalization capabilities of the model. Note that it is not possible to have disjoint positive samples due

to their sparsity (e.g. 4×4 pictures only have 30 positive images), thus we expect the models to correctly predict on all possible positive samples since they likely will have seen all of them. The real challenge of the task is to learn a model which correctly classifies all other images as negative. In addition to the mean squared error, we also report the classification accuracy of our model obtained via thresholding: the model classifies the image as positive if the predicted output is greater than 0.5 and classifies the image as negative otherwise. In all experiments, we initialize the tensor values with values independently drawn from a Gaussian distribution with mean zero and standard deviation 0.4. All models are trained with tensors of dimension $d = 6$ to allow the learning procedure to recover the automaton of Proposition 4.

Our first experiment considers the effect of varying training set sizes. Here, all pictures are of size 4×4, the size of the dataset is varied from $N = 500$ to $N = 50{,}000$, and the test set is of fixed size $N_{test} = 100$. We use a learning rate of 0.01 and mini-batches of size 100. The training errors, testing errors, and classification accuracies for different values of N are reported in Table 1 and a plot of the mean squared error (MSE) at each iteration for the case $N = 10{,}000$ is shown in Figure 6. We observe that even for small training set sizes, the model is able to reduce the MSE and attain perfect (or near-perfect) classification accuracy after seeing a relatively small number of mini-batches, although the training and testing errors do increase as N decreases.
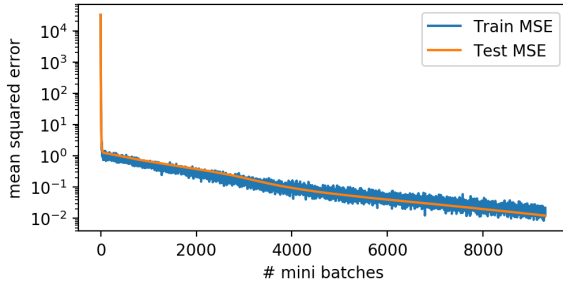
N	Training error	Testing error	Accuracy
50,000	$4.5 \cdot 10^{-4}$	$7.6 \cdot 10^{-4}$	100%
25,000	$7.5 \cdot 10^{-3}$	$9.4 \cdot 10^{-3}$	100%
10,000	$1.1 \cdot 10^{-2}$	$1.2 \cdot 10^{-2}$	100%
5,000	$6.0 \cdot 10^{-3}$	$7.9 \cdot 10^{-3}$	100%
1,000	$2.96 \cdot 10^{-3}$	$1.4 \cdot 10^{-2}$	100%
500	$2.4 \cdot 10^{-3}$	$2.2 \cdot 10^{-2}$	99%

Figure 6: MSE vs. number of mini-batches, when trained on 10,000 pictures of size 4×4. The final MSEs are reported in Table 1. Note that the MSE axis is log scaled.

Table 1: Training error, testing error, and classification accuracy for varying sizes of the training set.

Despite the low test and misclassification errors on 4×4 pictures, the learned models are unable to predict accurate values for pictures of larger size. For instance, when applied on a test set of 5×5 pictures, the GWM learned on the training set of size 50,000 yielded a mean squared error of 203.2 and an accuracy of 57% (recall that the data is a 50% split, so this accuracy is just slightly better than random). This is evidence that we have not recovered the GWM of proposition 4 (or an equivalent model computing the same function), since that automaton recognizes pictures of all sizes and thus would indeed have generalized.

The next experiment remedies this problem by combining multiple picture sizes together in the training set. Specifically, the training set is obtained by sampling 10,000 pictures (5,000 positives and 5,000 negatives) of different sizes ranging from $2 \times 2, 3 \times 3, \cdots, m \times m$. The test set contains 200 pictures of size $(m + 1) \times (m + 1)$. We run this
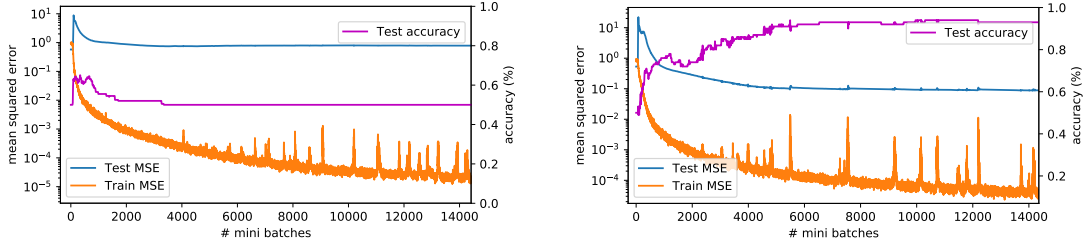
Figure 7: Training error, testing error and classification accuracy when training on pictures of sizes 2×2 up to 4×4 and testing on 5×5 pictures (left) and when training on pictures of sizes 2×2 up to 5×5 and testing on 6×6 pictures (right).

experiment for $m = 4$ and $m = 5$ and use a learning rate of 0.001 and mini-batches of size 1,000. The train and test MSE and the test accuracies are plotted as a function of the number of iterations in Figure 7 where we see that the learned model is able to generalize for the case of $m = 5$ but not for $m = 4$. The jump in performance observed from the case of $m = 4$ to the case of $m = 5$ is attributed to the greater wealth of data available in the training set. This experiment suggests that, given enough data, minimizing the mean squared error over the training data using gradient-based methods allows one to recover a GWM which is a good approximation of the target picture series.

4.2. Shifting Bits

Next, we investigate the performance of our methods on classification tasks with samples generated from the Shifting Bits language. In this language, despite the class imbalance, there are enough positive samples ($n2^n - 1$ out of 2^{mn}) so that the training and testing sets can be fully disjoint for both the positive and the negative samples. We initialize the tensor values with samples drawn from a Gaussian distribution with mean zero and standard deviation 0.2, and all models have dimension $d = 10$. Unlike the previous experiments, here the dimension is treated as a hyperparameter since we are not making use of a recognizability result. The learning rate was set to 0.01, with mini-batches of size 128, and we used gradient clipping due to the greater instability observed when minimizing the cross-entropy with the use of the sigmoid function.

We trained models on multiple datasets of size N consisting of pictures with fixed height m and different widths n, and tested the ability of the model to generalize to pictures of unseen sizes. Specifically, we experimented with datasets of sizes ranging from $N = 1000$ to $N = 20,000$, with one model trained on pictures of sizes $2 \times 5, 2 \times 6, \ldots, 2 \times 15$ and the other trained on pictures of sizes $3 \times 5, 3 \times 6, \ldots, 3 \times 15$. For each model, we tested on $N_{test} = 200$ pictures of widths $10, 20, 50$, and 100. The width of each picture in the training set was uniformly drawn from $5, \ldots, 15$. The testing accuracy for different pictures as a function of the training height and dataset size is given in Table 2 and a plot of the different testing accuracies at each epoch of training for the case of $m = 2, N = 5,000$ is shown in Figure 8.

Note that an epoch corresponds to a full pass through the training set (i.e. $N/128$ number of mini-batches). The figure shows that the model quickly converges to high accuracies on all $2 \times n$ sets, and correspondingly minimizes the cross-entropy errors (not shown).
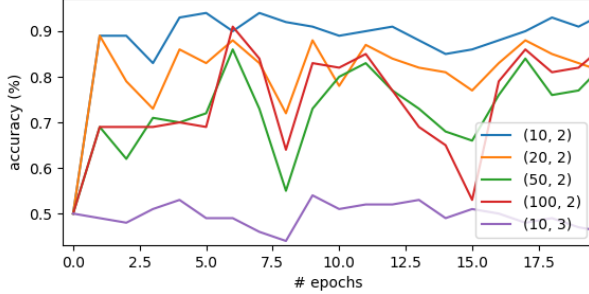

$m = 2, N =$	2×10	2×20	2×50	2×100	3×10
1000	67.0%	67.0%	61.0%	59.0%	61.0%
5000	94.0%	90.0%	73.0%	69.0%	58.0%
20000	100.0%	96.0%	96.0%	92.0%	53.0%
$m = 3, N =$	3×10	3×20	3×50	3×100	2×10
1000	49.0%	48.0%	60.0%	50.0%	58.0%
5000	45.0%	51.0%	50.0%	51.0%	48.0%
20000	100.0%	100.0%	97.0%	100.0%	50.0%

Figure 8: Accuracy vs. epochs when tested on different picture sizes and trained on 5,000 pictures of height 2 and widths uniformly drawn between $5, \ldots, 15$. The final accuracies are reported in Table 2.

Table 2: Testing accuracies on different picture sizes for varying dataset sizes, for models trained on pictures of height 2 (above) and height 3 (below).

Given enough data, the learned models for $m = 2$ and $m = 3$ are able to generalize to wider pictures of the same height, attaining accuracies greater than 90%, although for the case $m = 3$ more training data is needed to obtain accurate predictions. In addition to the training and test sets being disjoint, we note that the models are tested on pictures that can have shifts of size greater than 15 (while all training pictures had, at most, a shift of 15). However, we observe that the models trained on size $m = 2$ (resp. $m = 3$) are unable to correctly classify pictures of height 3 (resp. 2), and can only attain accuracies of around 50%.

In an effort to train a model which would generalize to multiple heights, we attempted to combine pictures of heights 2 and 3 (still with multiple widths) together in the training set, and to test on pictures of height 2, 3, and 4. However, unlike in the Bars & Stripes experiments, we found that it was possible to minimize the training error but to not correctly classify all the test sets. In our experiments the models would only settle for classifying a single picture size to near-perfect accuracies and attain 50% on other sizes. Some moderate but non-exhaustive exploration was done on the hyperparameters (i.e. learning rate and dimension of the model) with no significant impact on performance. We leave further experimentation for future work.

5. Conclusion

We investigated the use of gradient-based algorithms for learning GWMs from data and provided empirical evidence suggesting that it is possible to recover a GWM which approximates a function recognizing formal picture languages. We tackled regression and classification tasks and showed that in both cases the training and testing errors can be minimized leading to models which generalize to unseen pictures of different sizes. Our

experiments on the toy Bars & Stripes and Shifting Bits picture languages indicate that this is a promising direction, although this is just a first step. In the future, we intend to apply these methods to more challenging tasks, including more traditional machine learning problems which have typically not been viewed from the formal languages perspective. Furthermore, these methods can equally be applied to other families of graphs which could lead to interesting applications (e.g. in bioinformatics or in natural language processing). Moreover, a theoretical analysis of this approach remains to be done and is of particular interest. Extending the spectral learning algorithms to WPA and GWMs is also a direction that we will continue to investigate.

References

Parvaneh Babari and Manfred Droste. A Nivat theorem for weighted picture automata and weighted MSO logic. In *Language and Automata Theory and Applications*, volume 8977 of *LNCS*, pages 703–715. Springer, 2015.

Raphaël Bailly, François Denis, and Liva Ralaivola. Grammatical inference as a principal component analysis problem. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 33–40. ACM, 2009.

Raphaël Bailly, François Denis, and Guillaume Rabusseau. Recognizable series on hypergraphs. In *International Conference on Language and Automata Theory and Applications*, pages 639–651. Springer, 2015.

Raphaël Bailly, Guillaume Rabusseau, and François Denis. Recognizable series on graphs and hypergraphs. *Journal of Computer and System Sciences (in press)*, 2017.

Borja Balle, Xavier Carreras, Franco M Luque, and Ariadna Quattoni. Spectral learning of weighted automata. *Machine learning*, 96(1-2):33–63, 2014.

Jean Berstel and Christophe Reutenauer. Recognizable formal power series on trees. *Theoretical Computer Science*, 18(2):115–148, 1982.

Jean Berstel and Christophe Reutenauer. *Rational series and their languages*. Springer-Verlag Berlin, 1988.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

Symeon Bozapalidis and Archontia Grammatikopoulou. Recognizable picture series. *Journal of Automata, Languages and Combinatorics*, 10(2/3):159–183, 2005.

Ina Fichtner. Weighted picture automata and weighted logics. *Theory of Computing Systems*, 48(1):48–78, 2011.

Dora Giammarresi and Antonio Restivo. Two-dimensional finite state recognizability. *Fundamenta Informaticae*, 25(3, 4):399–422, 1996.

Dora Giammarresi, Antonio Restivo, Sebastian Seibert, and Wolfgang Thomas. Monadic second-order logic over rectangular pictures and recognizability by tiling systems. *Information and computation*, 125(1):32–45, 1996.

Daniel J. Hsu, Sham M. Kakade, and Tong Zhang. A spectral algorithm for learning hidden markov models. In *Proceedings of the Conference on Learning Theory*, 2009.

Katsushi Inoue and Akira Nakamura. Nonclosure properties of two-dimensional on-line tessellation acceptors and one way parallel sequential array acceptors. *IEICE TRANS-ACTIONS (1976-1990)*, 60(9):475–476, 1977.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Michel Latteux and David Simplot. Recognizable picture languages and domino tiling. *Theoretical computer science*, 178(1):275–283, 1997.

David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Ina Mäurer. Recognizable and rational picture series. In *Conference on Algebraic Informatics*, pages 141–155. Citeseer, 2005.

Jan Melchior, Asja Fischer, and Laurenz Wiskott. How to center deep boltzmann machines. *The Journal of Machine Learning Research*, 17(1):3387–3447, 2016.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Robert NC Pfeifer, Jutho Haegeman, and Frank Verstraete. Faster identification of optimal contraction sequences for tensor networks. *Physical Review E*, 90(3):033315, 2014.

Guillaume Rabusseau. *A Tensor Perspective on Weighted Automata, Low-Rank Regression and Algebraic Mixtures*. PhD thesis, Aix-Marseille Université, 2016.

Guillaume Rabusseau. Minimization of graph weighted models over circular strings. In *International Conference on Foundations of Software Science and Computation Structures*, pages 513–529. 2018.

Marcel Paul Schützenberger. On the definition of a family of automata. *Information and control*, 4(2-3):245–270, 1961.