Neural Network Based Nonlinear Weighted Finite Automata

Tianyu Li
School of Computer Science
McGill University
Montreal, Quebec H3A 2A7
Email: tianyu.li@mail.mcgill.ca

Guillaume Rabusseau
School of Computer Science
McGill University
Montreal, Quebec H3A 2A7
Email: guillaume.rabusseau@mail.mcgill.ca

Doina Precup
School of Computer Science
McGill University
Montreal, Quebec H3A 2A7
Email: dprecup@cs.mcgill.ca

***Abstract*—Weighted finite automata (WFA) can expressively model functions defined over strings. However, a WFA is inherently a linear model. In this paper, we propose a neural network based nonlinear WFA model along with a learning algorithm. Our learning algorithm performs a *nonlinear* decomposition of the so-called Hankel matrix (using an encode-decoder neural network) from which the transition operators of the model are recovered. We assessed the performance of the proposed model in a simulation study.**

1. Introduction

Many tasks in natural language processing, computational biology, or reinforcement learning, rely on estimating functions mapping sequences to real numbers. *Weighted finite automata (WFA)* are finite state machines that allow one to succinctly represents such functions. WFA has been widely used in many fields such as grammatical parsing [Mohri and Pereira, 1998], sequence modeling and prediction [Cortes et al., 2004] and bioinfomatics [Allauzen et al., 2008]. A *probabilistic WFA (PFA)* is a WFA satisfying some constraints and computes a probability distribution over strings. PFA is expressively equivalent to *Hidden markov models (HMM)* which have achieved success in many tasks such as speech recognition [Gales and Young, 2008] and human activity recognition [Nazábal and Artés-Rodríguez, 2015]. Recently, the so-called *spectral method* has been proposed as an alternative to EM based algorithms to learn HMM [Hsu et al., 2009], WFA [Bailly et al., 2009], predictive state representations [Boots et al., 2011], and related models. Compared to EM, the spectral method has the benefits of providing consistent estimators and reducing computational complexity.

Neural networks date back to the 1950's and have recently led to state-of-art results in practical applications, among others in computer vision and reinforcement learning. For example, deep convolutions neural networks [Krizhevsky et al., 2012] have pushed the boundary of computer vision to a new level. Furthermore the deep Q network architecture [Mnih et al., 2013], used as a function approximator for the action value function in reinforcement learning, has helped AI reach the level of professional players in the game of go.

In this paper, we propose a nonlinear WFA model based on neural networks, along with a learning algorithm. This model can be seen as an extension of dynamical recognizers [Moore, 1997] — which are in some sense a nonlinear extension of deterministic finite automata — to the quantitative setting. In contrast with recurrent neural networks, our learning algorithm does not rely on back-propagation through time. Instead, akin to the spectral method for WFAs, our learning algorithm performs a nonlinear factorization of the so-called Hankel matrix, which is then leveraged to recover the transition operators of the nonlinear WFA.

2. Preliminaries

We first introduce notions on weighted automata and the spectral learning method.

Weighted finite automaton. Let Σ^* denote the set of strings over a finite alphabet Σ. Denote the empty word by λ. A *weighted finite automaton* (WFA) with k states is a tuple $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}\rangle$ where $\boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty \in \mathbb{R}^k$ are the initial and final weight vector respectively, and $\mathbf{A}_\sigma \in \mathbb{R}^{k\times k}$ is the transition matrix for each symbol $\sigma \in \Sigma$. A WFA computes a function $f_A : \Sigma^* \to \mathbb{R}$ defined for each word $x = x_1x_2\cdots x_n \in \Sigma^*$ by

$$f_A(x) = \boldsymbol{\alpha}_0^\top \mathbf{A}_{x_1}\mathbf{A}_{x_2}\cdots\mathbf{A}_{x_n}\boldsymbol{\alpha}_\infty.$$

By letting $\mathbf{A}_x = \mathbf{A}_{x_1}\mathbf{A}_{x_2}\cdots\mathbf{A}_{x_n}$ for any word $x = x_1x_2\cdots x_k \in \Sigma^*$ we will often use the shorter notation $f_A(x) = \boldsymbol{\alpha}_0^\top \mathbf{A}_x\boldsymbol{\alpha}_\infty$. A WFA A with k states is *minimal* if its number of states is minimal, i.e., any WFA B such that $f_A = f_B$ has at least k states. A function $f : \Sigma^* \to \mathbb{R}$ is *recognizable* if it can be computed by a WFA. In this case the *rank* of f is the number of states of a minimal WFA computing f. If f is not recognizable we let $\operatorname{rank}(f) = \infty$.

Hankel matrix. The *Hankel matrix* $\mathbf{H}_f \in \mathbb{R}^{\Sigma^*\times\Sigma^*}$ associated with a function $f : \Sigma^* \to \mathbb{R}$ is the bi-infinite matrix with entries $(\mathbf{H}_f)_{u,v} = f(uv)$ for all words $u, v \in \Sigma^*$. The spectral learning algorithm for WFAs relies on the following fundamental relation between the rank of f and the rank of the Hankel matrix $\mathbf{H}_f$ [Carlyle and Paz, 1971, Fliess, 1974]:

Theorem 2.1. For any $f : \Sigma^* \to \mathbb{R}$, $\operatorname{rank}(f) = \operatorname{rank}(\mathbf{H}_f)$.

In practice, one deals with finite sub-blocks of the Hankel matrix. Given a basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$, where $\mathcal{P}$ is a set of *prefixes* and $\mathcal{S}$ is a set of *suffixes*, we define the Hankel matrix $\mathbf{H}_\mathcal{B} \in \mathbb{R}^{\mathcal{P}\times\mathcal{S}}$. Among all possible basis, we are particularly interested in the ones with the same rank as f. We say that a basis is *complete* if $rank(\mathbf{H}_\mathcal{B}) = rank(f) = rank(\mathbf{H}_f)$. Now, for an arbitrary basis $\mathcal{B} = (\mathcal{P}, \mathcal{S})$, define its *p-closure* as $\mathcal{B}' = (\mathcal{P}', \mathcal{S})$, where $\mathcal{P}' = \mathcal{P} \cup \Sigma \cup \{\lambda\}$. It turns out that a Hankel matrix over a p-closed basis can be partitioned into $|\Sigma| + 1$ blocks of the same size [Balle et al., 2014]:

$$\mathbf{H}_{\mathcal{B}'}^\top = [\mathbf{H}_\lambda^\top | \mathbf{H}_{\sigma_1}^\top | \cdots | \mathbf{H}_{\sigma_{|\Sigma|}}^\top]$$

where for each $\sigma \in \Sigma \cup \{\lambda\}$ the matrix $\mathbf{H}_\sigma \in \mathbb{R}^{\mathcal{P}\times\mathcal{S}}$ is defined by $\mathbf{H}_\sigma(u, v) = f(u\sigma v)$.

Spectral learning. It is easy to see that the rank of the Hankel matrix $\mathbf{H}_f$ is upper bounded by the rank of f: if $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}\rangle$ is a WFA with k states computing f, then $\mathbf{H}_f$ admits the rank k factorization $\mathbf{H}_f = \mathbf{PS}$ where the matrices $\mathbf{P} \in \mathbb{R}^{\Sigma^*\times k}$ and $\mathbf{S} \in \mathbb{R}^{k\times\Sigma^*}$ are defined by $\mathbf{P}_{u,:} = \boldsymbol{\alpha}_0^\top \mathbf{A}^u$ and $\mathbf{S}_{:,v} = \mathbf{A}^v \boldsymbol{\alpha}_\infty$ for all $u, v \in \Sigma^*$. Also, one can check that $\mathbf{H}_\sigma = \mathbf{P}\mathbf{A}_\sigma\mathbf{S}$ for each $\sigma \in \Sigma$. The spectral learning algorithm relies on the non-trivial observation that this construction can be reversed: given any rank k factorization $\mathbf{H}_\lambda = \mathbf{PS}$, the WFA $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}\rangle$ defined by $\boldsymbol{\alpha}_0^\top = \mathbf{P}_{\lambda,:}$, $\boldsymbol{\alpha}_\infty = \mathbf{S}_{:,\lambda}$ and $\mathbf{A}_\sigma = \mathbf{P}^+\mathbf{H}_\sigma\mathbf{S}^+$, is a minimal WFA computing f [Balle et al., 2014, Lemma 4.1], where $\mathbf{H}_\sigma$ for $\sigma \in \Sigma \cup \lambda$ denotes the finite matrices defined above for a prefix closed complete basis $\mathcal{B}$.

3. Introducing Non-Linearity in WFA

The WFA model assumes that the transition operators $\mathbf{A}_\sigma$ are linear. It is natural to wonder whether this linear assumption sometimes induces a too strong model bias. Also, even for functions recognizable by linear WFAs, introducing non-linearity can potentially reduce the number of states. Consider the following example: given a WFA $A = \langle \boldsymbol{\alpha}_0, \boldsymbol{\alpha}_\infty, \{\mathbf{A}_\sigma\}\rangle$, the function $(f_A)^2 : u \mapsto f_A(u)^2$ is recognizable and can be computed by the WFA $A' = \langle \boldsymbol{\alpha}_0', \boldsymbol{\alpha}_\infty', \{\mathbf{A}_\sigma'\}\rangle$ with $\boldsymbol{\alpha}_0' = \boldsymbol{\alpha}_0 \otimes \boldsymbol{\alpha}_0$, $\boldsymbol{\alpha}_\infty' = \boldsymbol{\alpha}_\infty \otimes \boldsymbol{\alpha}_\infty$ and $\mathbf{A}_\sigma' = \mathbf{A}_\sigma \otimes \mathbf{A}_\sigma$, where $\otimes$ denotes Kronecker product. One can check that if $\mathrm{rank}(f_A) = k$, then $\mathrm{rank}(f_{A'})$ can be as large as k^2, but intuitively the *true dimension* of the model is k using non-linearity[1]. These two observations motivate us to introduce *nonlinear WFAs*.

3.1. Nonlinear Weighted Finite Automata

Let us now use the notation $\tilde{g}$ to stress that a function g may be nonlinear. We define a *nonlinear WFA* $\tilde{A}$ of size k as a tuple $\langle \boldsymbol{\alpha}_0, \tilde{G}_\lambda, \{\tilde{G}_\sigma\}_{\sigma\in\Sigma}\rangle$, where $\boldsymbol{\alpha}_0 \in \mathbb{R}^k$ is a vector of initial weights, $\tilde{G}_\sigma : \mathbb{R}^k \to \mathbb{R}^k$ is a transition function for each $\sigma \in \Sigma$ and $\tilde{G}_\lambda : \mathbb{R}^k \to \mathbb{R}$ is a termination function.

1. By applying the spectral method on the component-wise square root of the Hankel matrix of A', one would recover the WFA A of rank k.

A nonlinear WFA $\tilde{A}$ computes a function $f_{\tilde{A}} : \Sigma^* \to \mathbb{R}$ defined by

$$f_{\tilde{A}}(x) = \tilde{G}_\lambda(\tilde{G}_{x_t}(\cdots \tilde{G}_{x_2}(\tilde{G}_{x_1}(\boldsymbol{\alpha}_0))\cdots))$$

for any word $x = x_1x_2\cdots x_t \in \Sigma^*$. This nonlinear model can be seen as generalization of dynamical recognizers [Moore, 1997] to the quantitative setting. Of course some restrictions on the nonlinear functions $\tilde{G}_\sigma$ have to be imposed in order to control the expressiveness of the model. In this paper, we consider nonlinear functions computed by multilayer perceptrons.

3.2. A Representation Learning Perspective on the Spectral Algorithm

Before presenting our learning algorithm for nonlinear WFAs, we first give some intuitions by showing how the spectral method in the linear case can be interpreted as a representation learning scheme. The spectral method can be summarized as a two stage process consisting of a *factorization step* and a *regression step*: first find a low rank factorization of the Hankel matrix and then perform regression to estimate the transition operators $\{\mathbf{A}_\sigma\}_{\sigma\in\Sigma}$.

In the factorization step, the rank k factorization $\mathbf{H}_f = \mathbf{PS}$ can be seen as finding a low dimensional feature space $\mathbb{R}^k$ where each prefix $u \in \mathcal{P}$ is mapped to a (row) feature vector $\phi(u) = \mathbf{P}_{u,:} \in \mathbb{R}^k$, in such a way that $\phi(u)$ encodes all the information sufficient to predict the value $f(uv)$ for any suffix $v \in \mathcal{S}$: indeed $f(uv) = \phi(u)\mathbf{S}_{:,v}$. In this process, we can view $\mathbf{P}$ and $\mathbf{S}$ as an encoder-decoder framework. By applying the encoder $\mathbf{P}$ to the prefix u, we are able to obtain a latent space embedding $\phi(u)$ from which the decoder $\mathbf{S}$ is able to recover the vector $\mathbf{H}_{u,:} \in \mathbb{R}^\mathcal{S}$ that contains all the values computed by the WFA on uv for arbitrary suffixes $v \in \mathcal{S}$. In the probabilistic setting, this vector would contain the probabilities of all suffixes after observing u. Thus, with this framework in mind, it seems natural to consider applying a nonlinear encoder-decoder instead of the linear factorization.

For regression step, one recovers the matrices $\mathbf{A}_\sigma$ satisfying $\mathbf{H}_\sigma = \mathbf{P}\mathbf{A}_\sigma\mathbf{S}$. From our encoder-decoder perspective, this can be seen as recovering the compositional mapping $\mathbf{A}_\sigma$ satisfying $\phi(u\sigma) = \phi(u)\mathbf{A}_\sigma$ for each $\sigma \in \Sigma$. This provides us with another way of applying non-linearity: assuming that the transition functions are nonlinear, we have $\phi(u\sigma) = \tilde{G}_\sigma(\phi(u))$, meaning that the compositional maps allowing us to obtain the embedding of a word $u\sigma$ given the embedding of the word u can now be nonlinear.

Going back to the problem of learning nonlinear WFAs, suppose we have an encoder $\phi_P : \mathbb{R}^\mathcal{P} \to \mathbb{R}^k$ and a decoder $\phi_S : \mathbb{R}^k \to \mathbb{R}^\mathcal{S}$ satisfying $\phi_S(\phi_P(\mathbf{u})) = \mathbf{u}^\top\mathbf{H}_\lambda$ for all $u \in \mathcal{P}$, where $\mathbf{u}$ denotes the one-hot encoding of the prefix u. This encoder-decoder framework ensures that there exists a k-dimensional representation space for prefixes for which the information encoded by ϕ_P is enough to predict any computation after observing a given prefix. The task of recovering the nonlinear compositional maps then consists in

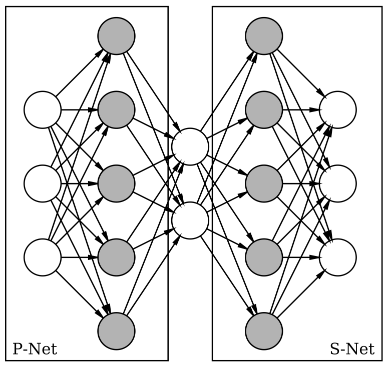

Figure 1: Nonlinear factorization, where grey units indicating nonlinear units while white ones are linear units.

finding $\tilde{G}_\sigma : \mathbb{R}^k \to \mathbb{R}^k$ such that $\phi_S(\tilde{G}_\sigma(\phi_P(\mathbf{u}))) = \mathbf{u}^\top \mathbf{H}_\sigma$ for any prefix $u \in \mathcal{P}$.

3.3. Learning Nonlinear WFAs

By now we have given intuition on the motivations for applying non-linearity in the factorization and regression steps. Introducing non-linearity can be achieved in several ways. In this paper, we will use neural networks due to their strong power as function approximators and their ability to discover relevant low-dimensional representation spaces.

Nonlinear factorization. As observed in the previous paragraphs, incorporating non-linearity in the factorization step can be achieved by finding an encoder $\phi_P(\cdot)$ and a decoder $\phi_S(\cdot)$ satisfying $\phi_S(\phi_P(\mathbf{x})) = \mathbf{x}^\top \mathbf{H}_\lambda$ for all $\mathbf{x}$. To do this, we propose the neural net model shown in Figure 1.

Assume $\mathbf{H}_\lambda \in \mathbb{R}^{m\times n}$, the model is trained to map random inputs $\mathbf{x} \in \mathbb{R}^m$ to $\mathbf{x}^\top \mathbf{H}_\lambda \in \mathbb{R}^n$. This is achieved by minimizing the mean squared error between the true value and the predicted value. Instead of linearly factorizing the Hankel matrix, we use two neural nets: P-Net and S-Net, whose hidden layer activation functions are nonlinear[2]. Denote the nonlinear activation function by θ, and let $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$ be the weights matrices of the neural net shown in Figure 1 (from left to right). The function $\hat{f} : \mathbb{R}^m \to \mathbb{R}^n$ computed by the neural net can be written as

$$\hat{f} = \tilde{f}_S \circ \tilde{f}_P : \mathbf{x} \mapsto \theta(\theta(\mathbf{x}\mathbf{A})\mathbf{B}\mathbf{C})\mathbf{D}$$

where the encoder-decoder functions $\tilde{f}_P : \mathbb{R}^m \to \mathbb{R}^k$ and $\tilde{f}_S : \mathbb{R}^k \to \mathbb{R}^n$ are defined by $\tilde{f}_P(\mathbf{x}) = \theta(\mathbf{x}\mathbf{A})\mathbf{B}$ and $\tilde{f}_S(\mathbf{x}') = \theta(\mathbf{x}'\mathbf{C})\mathbf{D}$ for all row vectors $\mathbf{x} \in \mathbb{R}^m, \mathbf{x}' \in \mathbb{R}^k$.

If we let f_λ be the function represented by the Hankel matrix $\mathbf{H}_\lambda$ (i.e. $f_\lambda(\mathbf{x}) = \mathbf{x}^\top \mathbf{H}_\lambda$), we see that the network is trained to minimize the ℓ_2 distance between $\hat{f}$ and f_λ. It is easy to check that if the activation function is linear, one will exactly recover the rank factorization from the linear case with $\mathbf{P} = \mathbf{AB}$ and $\mathbf{S} = \mathbf{CD}$.

2. We use the (component-wise) tanh function in our experiments.

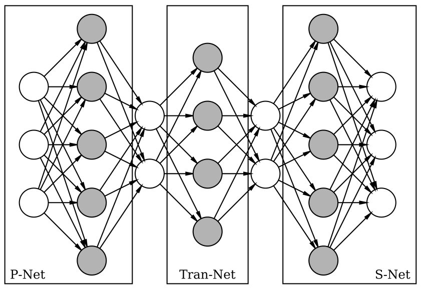

Figure 2: Network for the transition functions (grey layer indicates nonlinear units while white ones are linear).

Nonlinear regression. Given the encoder-decoder maps $\tilde{f}_S$ and $\tilde{f}_P$, we then move on to recovering the transition functions. Recall that we wish to find $\tilde{G}_\sigma : \mathbb{R}^k \to \mathbb{R}^k$ for each σ such that the embedding $\phi(u\sigma) = \tilde{f}_P(\mathbf{u}\boldsymbol{\sigma})$ can be obtained by applying $\tilde{G}_\sigma$ to $\phi(u)$ for any $\sigma \in \Sigma$ and any prefix u (where $\mathbf{u}\boldsymbol{\sigma}$ denotes the one-hot encoding of $u\sigma$). This can be achieved by enforcing $\tilde{f}_S(\tilde{G}_\sigma(\tilde{f}_P(\mathbf{u})))$ to be close to the vector containing the true values $f(u\sigma v)$ for all suffixes v, i.e. by minimizing the distance between $\tilde{f}_S(\tilde{G}_\sigma(\tilde{f}_P(\mathbf{u})))$ and $\mathbf{u}^\top \mathbf{H}_\sigma$ for all prefixes u. Using a neural network with one hidden layer for the function $\tilde{G}_\sigma$, which we refer to as Tran-Net, we obtain the neural network architecture shown in Figure 2, where the P-Net and the S-Net are the ones obtained from the previous factorization step. During training, the weights of the P-Net and of the S-Net are not updated and the weights of the Tran-Net are optimized to minimize the squared error between $\mathbf{x}^\top \mathbf{H}_\sigma$ and the output of the network for randomly generated inputs $\mathbf{x} \in \mathbb{R}^m$. Let $\mathbf{E}_\sigma, \mathbf{F}_\sigma$ be the two weights matrices in the Tran-net in Figure 2 (from left to right). The function $\tilde{G}_\sigma : \mathbb{R}^k \to \mathbb{R}^k$ it computes is defined by $\tilde{G}_\sigma(\mathbf{x}) = \mathbf{F}_\sigma \theta(\mathbf{E}_\sigma \mathbf{x})$. If we let f_σ be the function represented by the Hankel matrix $\mathbf{H}_\sigma$ (i.e. $f_\sigma(\mathbf{x}) = \mathbf{x}^\top \mathbf{H}_\sigma$), the network is trained to minimize the ℓ_2 distance between f_σ and $\tilde{f}_S \circ \tilde{G}_\sigma \circ \tilde{f}_P$.

Recovering the nonlinear WFA. Given the transition functions $\tilde{G}_\sigma$ obtained above, for the nonlinear WFA $\tilde{A}$ we defined, we still need to specify $\boldsymbol{\alpha}_0$ and $\tilde{G}_\lambda$. From the encoder-decoder point of view, the initial vector $\boldsymbol{\alpha}_0$ is the embedding of the empty word λ obtained through the encoder ϕ_P and the termination function $\tilde{G}_\lambda$ is the decoding function corresponding to the suffix λ. In particular, in the probabilistic setting $\tilde{G}_\lambda(\mathbf{u})$ can be seen as the probability of stopping the computation and emitting the word u. Therefore, we let $\boldsymbol{\alpha}_0 = \tilde{f}_P(\boldsymbol{\lambda})$ and $\tilde{G}_\lambda : \mathbf{x} \to \tilde{f}_S(\mathbf{x})^\top \boldsymbol{\lambda} \in \mathbb{R}$ (where $\boldsymbol{\lambda}$ is the one-hot encoding of the empty word).

4. Experiments

We compare the classical spectral learning algorithm with three configurations of our neural-net based method: applying non-linearity only in the factorization step (denoted by *fac.non*), only in the regression step (denoted by *tran.non*), and in both phases (denoted by *both.non*).

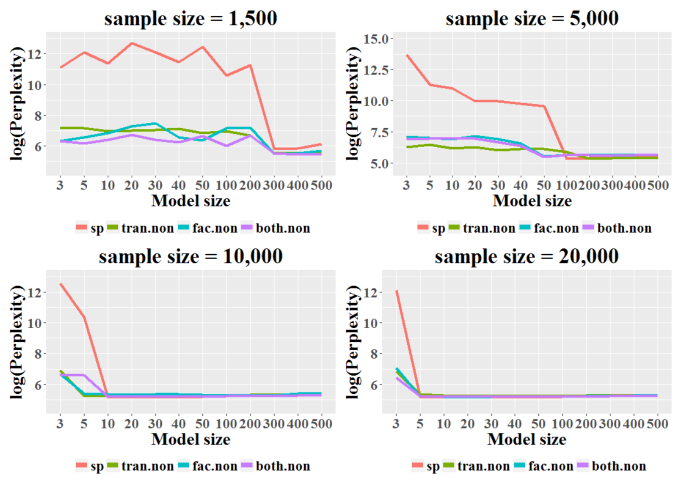

Figure 3: log(Perplexity) of Pautomac2 dataset based on different number of states and sample size. The data is generated from a HMM with 63 states and 18 symbols.

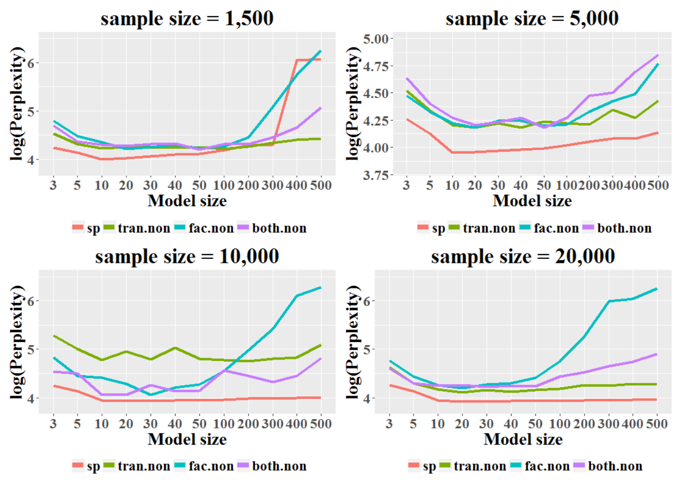

Figure 4: log(Perplexity) of Pautomac3 dataset based on different number of states and sample size. The data is generated from a PFA with 25 states and 4 symbols.

In the experiments, we use empirical frequencies in a training data set (of sizes ranging from $1,500$ to $20,000$) to estimate the sub block of the Hankel matrix $\mathbf{H}_{\mathcal{B}}$, where the basis $\mathcal{B}$ is obtained by selecting the first $1,000$ most frequent prefixes and suffixes in the training data. We compared the model for different model sizes k ranging from 1 to 500, where k is the number of states for the spectral method and the size of nonlinear WFAs. For the latter, the number of hidden units of P-Net and S-Net is set to $2,000$, while the number of hidden units in the Tran-Net is set to $2k$[3].

We used synthetic data from the Pautomac challenge [Verwer et al., 2014], and we evaluated the models using the perplexity measure on a test set T of size $1,000$ defined by $Perp(M) = -2^H$ which is normalized so that it sums to 1 according to Pautomac Challenge. Here we have $H = \sum_{x \in T} P_T(x) \log(P_M(x))$ and P_M, P_T denote the probabilities calculated by the model and the true probabilities respectively. We should keep in mind that all the datasets from Pautomac are generated from linear models, thus using a nonlinear model might suffer from model mismatch. We used problems 2 and 3 form the Pautomac challenge and the results are reported in Figure 3 and Figure 4 respectively. For pautomac2, we see that when the sample size is relatively small, such as 1500 and 5000, the original linear spectral learning will be outperformed by our methods for small model sizes. When the sample size increases, even with extreme small model size, nonlinear models still show reasonable perplexity indicating their potential benefits over linear models. This confirms our intuition that applying non-linearity can be beneficial when dealing with small number of states: in the Kronecker product example, although the transition functions are linear they admit a nonlinear lower dimensional representation. Therefore, by using nonlinear models, one can discover this underlying nonlinear structure and thus potentially reduce the model size. For small sample sizes, spectral learning tends to exhibit worse performance than our models (especially for small model sizes). This is counter-intuitive but we suspect that the linear model is affected by noise in the training stage in this case. However, this requires further investigation.

The results for pautomac2 are for a relatively complicated model in terms of number of states and actions. However, when dealing with a simpler model, such as pautomac3, our experiments show that non-linearity in this case could be detrimental. From Figure 4, we can see that for every sample size spectral learning outperforms our model. This is somehow not surprising, as the underlying structure of the model is linear and easier to capture. However, more interestingly, this experiment exhibits an overfitting behavior that was not present in the previous experiment due to the simplicity of the underlying model. In general, nonlinear factorization is the one that suffers most from overfitting as shown in the graph, while nonlinear regression and applying non-linearity in both stages are less prone to overfitting and are relatively steady even compared to spectral learning.

3. These hyper parameters are not finely tunes, thus some optimizations might potentially improve the results.

5. Conclusion and future work

In this paper, we defined a model of nonlinear weighted finite automata along with a learning algorithm. Our learning algorithm can be seen as applying non-linearity into the factorization and regression steps of the spectral learning algorithm. Empirical results showed that our model can perform better for complicated intrinsic structures and can cope with restricted number of states and sample sizes.

In future works, we intend to further assess the benefits of introducing non-linearity in WFAs on both nonlinear synthetic data and real word data, and to compare our approach with recurrent neural nets. Theoretically, it will be interesting to investigate the properties of the nonlinear WFA model from a formal language perspective.

Acknowledgments

Tianyu Li and Doina Precup have been supported by NSERC.

References

Cyril Allauzen, Mehryar Mohri, and Ameet Talwalkar. Sequence kernels for predicting protein essentiality. In *Proceedings of the 25th international conference on Machine learning*, pages 9–16. ACM, 2008.

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.

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

Byron Boots, Sajid M Siddiqi, and Geoffrey J Gordon. Closing the learning-planning loop with predictive state representations. *The International Journal of Robotics Research*, 30(7):954–966, 2011.

Jack W. Carlyle and Azaria Paz. Realizations by stochastic finite automata. *Journal of Computer and System Sciences*, 5(1):26–40, 1971.

Corinna Cortes, Patrick Haffner, and Mehryar Mohri. Rational kernels: Theory and algorithms. *Journal of Machine Learning Research*, 5(Aug):1035–1062, 2004.

Michel Fliess. Matrices de hankel. *Journal de Mathmatiques Pures et Appliques*, 53(9):197–222, 1974.

Mark Gales and Steve Young. The application of hidden markov models in speech recognition. *Foundations and trends in signal processing*, 1(3):195–304, 2008.

Daniel Hsu, Sham M Kakade, and Tong Zhang. A spectral algorithm for learning hidden markov models. In *Proceedings of the 22nd conference on learning theory*, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Mehryar Mohri and Fernando CN Pereira. Dynamic compilation of weighted context-free grammars. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 2*, pages 891–897. Association for Computational Linguistics, 1998.

Cristopher Moore. Dynamical recognizers: Real-time language recognition by analog computers. In *Foundations of Computational Mathematics*, pages 278–286. Springer, 1997.

Alfredo Nazábal and Antonio Artés-Rodríguez. Discriminative spectral learning of hidden markov models for human activity recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 1966–1970. IEEE, 2015.

Sicco Verwer, Rémi Eyraud, and Colin De La Higuera. Pautomac: a probabilistic automata and hidden markov models learning competition. *Machine learning*, 96(1-2): 129–154, 2014.