1 Supplement: Recurrent neural networks and automata

In this supplement, we outline how it is possible to derive all the recursions in the main paper directly, by combining state machines using one relatively simple operation. While this requires some additional notation up-front, the payoff is that complex recursions (such as those for α, β and γ) can be verified systematically (and algorithmically). Furthermore, concepts such as the "alignment envelope" have straightforward interpretations in this automata-centric framework.

1.1 Overview

Since the RNN's output $\mathbf{y}$ is like the parameterization of a gapped position-specific weight matrix, we can represent it using a profile Hidden Markov Model, i.e. a state machine with (probabilistically) weighted transitions [1]. This is illustrated in Figure 1. Let us denote this HMM by $\mathcal{R}(\mathbf{y})$.

Suppose ℓ represents a label sequence that may be the output of the RNN. One way to represent this constraint—that a sequence must have a particular value—is by using a second state machine as an indicator function that assigns unit weight to sequence ℓ, and zero weight to anything else. Denote this second state machine by $\mathcal{S}(\ell)$.

To impose the constraint on the HMM, we take the pointwise product of these two state machines (Figure 2) and find the norm, i.e. the sum over all path weights. This leads to various results:

- The probability that the RNN's output decodes to sequence ℓ is $|\mathcal{R}(\mathbf{y})\mathcal{S}(\ell)|$ and this probability takes time $\mathcal{O}(T|\ell|)$ to compute
- The probability that it decodes to a sequence whose prefix is κ is $|\mathcal{R}(\mathbf{y})\mathcal{P}(\kappa)|$, where $\mathcal{P}(\kappa)$ denotes $\mathcal{S}(\kappa)$ concatenated with a "wildcard" matcher
- The probability that two RNNs with outputs $\mathbf{y}, \mathbf{z}$ decode to the same sequence is $|\mathcal{G}|$ where $\mathcal{G} = \mathcal{R}(\mathbf{y})\mathcal{R}(\mathbf{z})$
- Calculations involving $\mathcal{G}$ take time and memory $\mathcal{O}(T^2)$ which is expensive, but admits optimizing approximations e.g. using a slimmed-down version of $\mathcal{G}$ that retains only the highest-probability alignments
- The probability that the consensus decoded sequence is ℓ is $|\mathcal{G}\mathcal{S}(\ell)|$, the probability that the decoded consensus has prefix κ is $|\mathcal{G}\mathcal{P}(\kappa)|$, and so on

In what follows, we describe some of the above probability calculations in more detail. We first review the CTC decoding approach in more detail. We then make the automata-theoretic terminology and lemmas more precise. Finally we outline dynamic programming recursions for decoding consensus labelings of paired RNNs.

1.2 Weighted Finite State Automata

A Weighted Finite State Automaton (informally, an "automaton" or "machine") over a semiring $\mathbb{W}$ of weights can be defined as a 5-tuple $\mathcal{A} = (Q, \Sigma, I, F, E)$, where:

- Q is a finite set, the set of states;
- Σ is a finite set, called the alphabet;
- I is a member of Q, the initial state;
- F is a member of Q, the final state;
- $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times Q \times \mathbb{W}$ (where ϵ is the empty string) is the finite set of (weighted, labeled) transitions.

For our purposes, the alphabet will generally be the label alphabet L of the recurrent neural network described earlier, and $\mathbb{W}$ will be the probability semiring.

Behavior. Define the weight of a path through the (Q, E) transition graph to be the $\mathbb{W}$-product of the transition weights. Similarly, define the label of the path to be the concatenation of its Σ-labels. For sequence $x \in \Sigma^*$ define the behavior, $[\mathcal{A}]_x$, to be the summed weight of all paths from I to F having label x.

Norm. Define the norm of $\mathcal{A}$ to be the sum over all paths from I to F

$$|\mathcal{A}| = \sum_x [\mathcal{A}]_x$$

Complexity. The norm can be computed in time $\mathcal{O}(|E|)$ and memory $\mathcal{O}(|Q|)$.

(Complexity results and other lemmas are outlined informally, for brevity. More rigorous treatments may be obtained by straightforward comparison with the results for transducer theory [2–4].)

Intersection. If $\mathcal{A}$ and $\mathcal{B}$ are both automata, there exists an automaton $\mathcal{AB}$ with behavior

$$[\mathcal{AB}]_x = [\mathcal{A}]_x [\mathcal{B}]_x$$

for all sequences x, and

$$|Q_{\mathcal{AB}}| = \mathcal{O}(|Q_{\mathcal{A}}| \cdot |Q_{\mathcal{B}}|)$$
$$|E_{\mathcal{AB}}| = \mathcal{O}(|E_{\mathcal{A}}| \cdot |E_{\mathcal{B}}|)$$

Kronecker delta. For any given sequence y, there exists an automaton $\mathcal{S}(y)$ with behavior

$$[\mathcal{S}(y)]_x \;=\; \delta(x = y)$$

An example construction of this automaton follows. Suppose that $y \in L^*$ is a sequence over some label alphabet L. Then

$$\mathcal{S}(y) = (Q_{\mathcal{S}(y)}, L, \epsilon, y, E_{\mathcal{S}(y)})$$

where

- the state space $Q_{\mathcal{S}(y)}$ is the set of prefixes of y, including y itself and the empty string ϵ;
- the alphabet is the label alphabet L;
- the initial state is the empty string ϵ;
- the final state is y;

– the transition set is $E_{\mathcal{S}(y)} = \{(p, a, q, 1) : p, q \in Q_{\mathcal{S}(y)}, a \in L, p \oplus a = q\}$ wherein $p \oplus a$ denotes the concatenation of p with a.

Clearly,

$$|Q_{\mathcal{S}(y)}| = |y| + 1$$
$$|E_{\mathcal{S}(y)}| = |y|$$

Linear algebra analogy. If an automaton $\mathcal{A}$ represents a vector then the behavior $[\mathcal{A}]_x$ (for a given sequence x) is an individual vector element; the norm $|\mathcal{A}|$ is the usual 1-norm; $\mathcal{S}(x)$ is a unit vector in the x'th dimension; and intersection $\mathcal{AB}$ is the pointwise product [3, 4].

The term *finite-state transducer* (vs finite-state automaton) is used to describe what, in the linear algebra analogy, would be a *matrix*. Rather than having a single alphabet, a transducer $\mathcal{T}$ has both an input alphabet and an output alphabet. It represents an operation that relates or aligns two sequences x, y, with a behavior that can be written $[\mathcal{T}]_{xy}$. If this behavior is appropriately normalized, it may represent the probabilistic transition matrix of a machine that stochastically transforms an input x into an output y. The automata-theoretic notation we use here is directly based on the standard transducer notation; however, for the algorithms described in this paper, no sequence transformation (*per se*) is being modeled, so we prefer the simpler vector notation (automata) over matrix notation (transducers).

Hidden Markov model analogy. An HMM may be represented as an automaton $\mathcal{A}$ with the norm constraint $|\mathcal{A}| = 1$, for probabilistic normalization. The behavior $[\mathcal{A}]_x$ represents the Forward likelihood for sequence x.

Concatenation. If $\mathcal{A}$ and $\mathcal{B}$ are automata, there exists an automaton $\mathcal{A} \oplus \mathcal{B}$ with behavior

$$[\mathcal{A} \oplus \mathcal{B}]_x = \sum_{\substack{y,z: \\ x=y \oplus z}} [\mathcal{A}]_y [\mathcal{B}]_z$$

for all sequences x, and

$$|Q_{\mathcal{A}\oplus\mathcal{B}}| = \mathcal{O}(|Q_{\mathcal{A}}| + |Q_{\mathcal{B}}|)$$
$$|E_{\mathcal{A}\oplus\mathcal{B}}| = \mathcal{O}(|E_{\mathcal{A}}| + |E_{\mathcal{B}}|)$$

Proofs. Constructions of intersected and concatenated automata, validating the given bounds, are straightforward by comparison with the corresponding constructions for transducers [2–4].

Kleene closure. If L is a finite set (the label alphabet), then there exists an automaton $\mathcal{C}(L)$ with behavior

$$[\mathcal{C}(L)]_x = 1 \quad \forall x \in L^*$$

An example construction of this automaton is

$$\mathcal{C}(L) = (\{q\}, L, q, q, E_{\mathcal{C}(L)})$$

where

– the state space contains the single state q;
– the alphabet is the label alphabet L;
– the initial state is q;
– the final state is q;
– the set of transitions is $E_{\mathcal{C}(L)} = \{(q, a, q, 1) : a \in L\}$

Conceptually, $\mathcal{C}(L)$ is like a wildcard in a regular expression: it matches anything. It may be regarded as the identity for automata intersection.

In terms of complexity,

$$|Q_{\mathcal{C}(L)}| = 1$$
$$|E_{\mathcal{C}(L)}| = |L|$$

Prefix automaton. For sequence $x \in L^*$, define the prefix automaton $\mathcal{P}(x)$ as the concatenation of the Kronecker delta $\mathcal{S}(x)$ with the Kleene closure $\mathcal{C}(L)$. Informally, $\mathcal{P}(x)$ accepts all sequences for which x is a prefix, assigning unit weight to all such sequences and zero weight to all other sequences.

RNN automaton. Suppose as before that a recurrent neural net has output sequence $\mathbf{y}$, length T. So y_a^t is the probability that position t of the output is labeled with a symbol a from the extended label alphabet $L' = L \cup \{\epsilon\}$.

Define the RNN automaton $\mathcal{R}(\mathbf{y}) = (\mathbb{Z}_{T+1}, L, 0, T+1, E_R(\mathbf{y}))$ where

– the state space is the set of integers modulo $T+1$, $\mathbb{Z}_{T+1} = \{0, 1 \ldots T\}$;
– the alphabet is the label alphabet L;
– the initial state is 0;
– the final state is T;
– the set of transitions is $E_R(\mathbf{y}) = \{(t, a, t+1, y_a^{t+1}) : a \in L', 0 \le t < T\}$.

Clearly,

$$|Q_{\mathcal{S}(y)}| = |E_{\mathcal{S}(y)}| = \mathcal{O}(T)$$

An RNN automaton for a small $\mathbf{y}$ matrix is shown in Figure 1.

Differentiability. This treatment can be extended to any set of probabilities $\mathbf{y}$ that specify a gapped position-specific weight matrix profile; it is not restricted to probabilities that are output by RNN classifiers. The crucial point, for deep learning purposes, is that if the y_a^t are differentiable functions of the parameters (and inputs), then the norms of automata constructed using the RNN automaton will also be differentiable. Furthermore, the transition graph structure of the automaton is easily related to the underlying architecture of the deep learning graph.

1.3 Automata-theoretic formulation of CTC decoding

The probability that a given sequence ℓ is the correct label sequence is the norm of an RNN automaton $\mathcal{R}(\mathbf{y})$ intersected with a Kronecker delta $\mathcal{S}(\ell)$:

$$\begin{aligned} P(\ell|\mathbf{x}) &= [\mathcal{R}(\mathbf{y})]_\ell \\ &= |\mathcal{R}(\mathbf{y})\mathcal{S}(\ell)| \end{aligned}$$

The probability that a given sequence κ is a prefix of the correct label sequence is the norm of the same RNN automaton $\mathcal{R}(\mathbf{y})$ intersected with a prefix automaton $\mathcal{P}(\kappa)$:

$$\sigma(\kappa|\mathbf{x}) = |\mathcal{R}(\mathbf{y})\mathcal{P}(\kappa)|$$

The states of $\mathcal{R}(\mathbf{y})\mathcal{S}(\ell)$ correspond closely to the Forward variables $\alpha_t^\epsilon(\ell_{1:s})$, $\alpha_t^*(\ell_{1:s})$ and Backward variables $\beta_t^\epsilon(\ell_{s:|\ell|})$, $\beta_t^*(\ell_{s:|\ell|})$ of Section **??**. Specifically, the Forward variables represent sums over paths that begin in the initial state, and end in each given state of the machine. Similarly, the Backward variables represent sums over paths from each state to the final state.

Efficient implementation of the prefix search algorithm relies on the fact that as the label ℓ is incrementally extended, the transition graph for automata such as $\mathcal{S}(\ell)$ and $\mathcal{R}(\mathbf{y})\mathcal{P}(\ell)$ can also be incrementally extended, as they will contain previously-computed transition graphs as subgraphs; so that many of the intermediate dynamic programming steps used to compute the behavior can be cached and re-used.

In the $1D^2$ decoding case with two output sequences $\mathbf{y}$, $\mathbf{z}$, the condition that the RNNs agree is again found by intersecting the automata, $\mathcal{G} = \mathcal{R}(\mathbf{y})\mathcal{R}(\mathbf{z})$. The label, prefix, and agreement probabilities are all norms of intersections

$$\begin{aligned} P(\ell|\mathbf{u},\mathbf{v}) &= [\mathcal{G}]_\ell \\ &= |\mathcal{G}\mathcal{S}(\ell)| \\ \sigma(\kappa|\mathbf{u},\mathbf{v}) &= |\mathcal{G}\mathcal{P}(\kappa)| \\ Z &= |\mathcal{G}\mathcal{C}(L)| \end{aligned}$$

The states of $\mathcal{G}$ and $\mathcal{G}\mathcal{S}(\ell)$ are related to the recursions for $\gamma_{u,v}^\epsilon$, $\gamma_{u,v}^{*\epsilon}$, $\gamma_{u,v}^{**}$ and $\alpha_{u,v}^\epsilon$, $\alpha_{u,v}^{*\epsilon}$, $\alpha_{u,v}^{**}$ in Section **??**. Restricting the alignment envelope to a subset $\mathbb{A}'$ of the full alignment envelope $\mathbb{A}$ is equivalent to using a smaller automaton $\mathcal{G}'$ whose transition graph is a subgraph of the full transition graph of $\mathcal{G}$ [3].

References

1. R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, Cambridge, UK, 1998.
2. M. Mohri, F. Pereira, and M. Riley. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16(1):69–88, 2002.
3. Oscar Westesson, Gerton Lunter, Benedict Paten, and Ian Holmes. Accurate reconstruction of insertion-deletion histories by statistical phylogenetics. *PLoS One*, 7(4):e34572, 2012.
4. A. Bouchard-Côté. A note on probabilistic models over strings: the linear algebra approach. *Bulletin of Mathematical Biology*, 75(12):2529–2550, 2013.

Fig. 1. Representation of Recurrent Neural Network outputs as Hiden Markov Model. The transition weights for the state machine $\mathcal{R}(\mathbf{y})$ (bottom) are given by the output probability matrix $\mathbf{y}$ of the RNN. The neural network may have any architecture, as long as the output matrix has the form shown here.

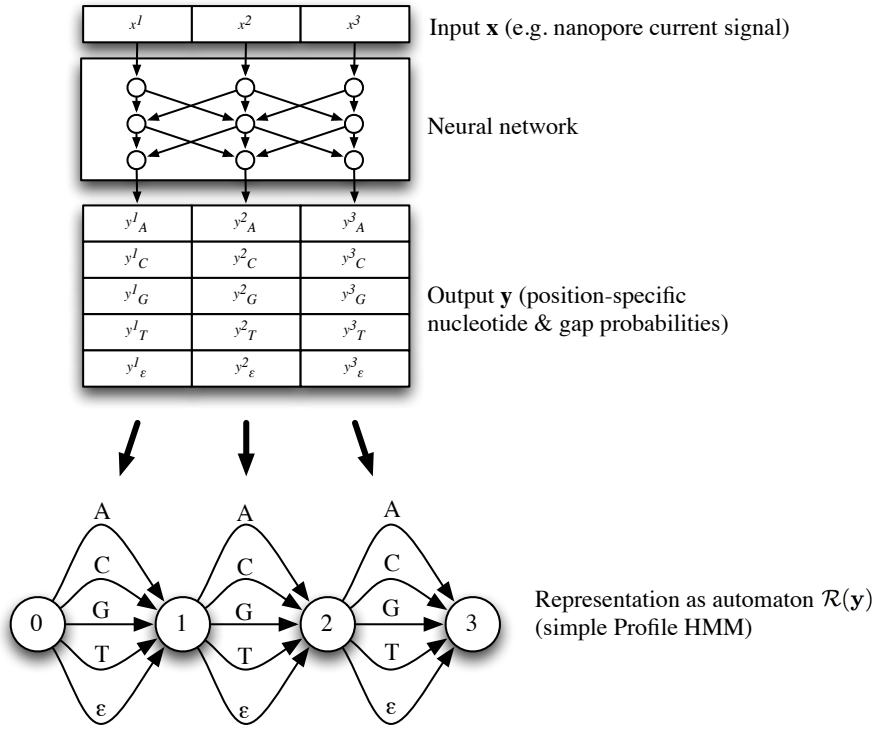

Fig. 2. Intersection of automata (analogous to pointwise vector multiplication) is used, among other things, to impose constraints on the sequence(s) that an HMM must emit. The intersection shown here is written $\mathcal{R}(\mathbf{y})\mathcal{S}(\texttt{AG})$ where $\mathcal{R}(\mathbf{y})$ represents the machine on the left, derived from a neural network (see Figure 1) and $\mathcal{S}(\texttt{AG})$ represents the machine in the middle, which allows only the sequence `AG`. Intersection is typically implemented by taking a Cartesian product of the two transition graphs, keeping only edges where the labels of the two machines are synchronized. Some states in the combined transition graph may be inaccessible (shown as grayed-out circles).