1 Definitions

So far, we have considered the Viterbi algorithm. It is an efficient method to compute the score of the best path in an ambiguous WFSA, but it has two drawbacks:

1. The variant we gave does not deal with \( \varepsilon \)-transitions.

2. It only supports one “aggregation” function—computing the maximum value of competing paths.

We wish to generalize the notion of path weights so as to be able to compute different quantities. Aggregating paths by maximizing or minimizing the weights of alternate computation paths in a machine is useful for finding the best sequence of states in a machine given some input; and summing over them lets you compute the total probability associated with some output. The algorithm we develop can also deal with \( \varepsilon \)-transitions, which the Viterbi algorithm we gave does not do.

These path weight aggregation computations (and many, many others) can be computed efficiently with just one algorithm provided that the weights, the aggregation function, and how path component weights “multiply” have certain properties. Namely, these must form a \textit{semiring}. A semiring is an algebraic structure consisting of:

- \( K \), a set (e.g., the real numbers, the natural numbers, \( \mathbb{R} \cup \{-\infty, +\infty\} \), ...);
- \( \oplus \), an addition operator that is associative and commutative (i.e., for all \( a, b, c \in K \)\(^3 \) associativity implies \( (a \oplus b) \oplus c = a \oplus (b \oplus c) \) and commutativity implies \( a \oplus b = b \oplus a \));
- \( \otimes \), a multiplication operator that is associative (i.e., for all \( a, b, c \in K \)\(^3 \), \( (a \otimes b) \otimes c = a \otimes (b \otimes c) \));
- \( 0 \in K \), an additive identity (i.e., \( 0 + a = a \) for all \( a \in K \) that is also an annihilator (i.e., \( 0 \otimes a = 0 \)); and
- \( 1 \in K \), a multiplicative identity (i.e., \( 1 \otimes a = a \) for all \( a \in K \)).

Additionally, \( \otimes \) must distribute over \( \oplus \), that is \( a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c) \) and \( (b \oplus c) \otimes a = (b \otimes a) \oplus (c \otimes a) \). If \( \oplus \) is commutative the semiring is said to be \textit{commutative}. If \( \oplus \) is idempotent (i.e., for every \( a \in K \), \( a \oplus a = a \)), then the semiring is said to be \textit{idempotent}. A semiring may have an additional \textit{closure operator} \( a^* \) which satisfies the axiom \( a^* = 1 \oplus a \otimes a^* \). A semiring equipped with a closure operator is said to be \textit{closed}.
1.1 A note on closure

There are generally two ways to think about the definition of the closure operator:

- In semirings where infinite series are well defined, \( a^* = \top \oplus a \oplus a^2 \oplus a^3 \oplus \cdots \);
- in semirings where \( \oplus \) has an inverse operation (subtraction) and reciprocals are defined, it is possible to define closure as \( a^* = (\top \ominus a)^{-1} \).

Both of these definitions fulfill the \( a^* = \top \oplus a \otimes a^* \) axiom.

1.2 Example: tropical semiring

\((\mathbb{R} \cup \{-\infty\}, \max, +, -\infty, 0)\) is a semiring.

1.3 Example: counting semiring

\((\mathbb{N} \cup \{+\infty\}, +, \times, 0, 1)\) is a semiring.

1.4 Example: Boolean semiring

\( (\{\text{true}, \text{false}\}, \lor, \land, \text{false}, \text{true}) \) is a semiring.

1.5 Example: probability semiring

\( (\mathbb{R}_{\geq 0} \cup \{+\infty\}, +, \times, 0, 1) \) is a semiring.

2 Generalized path sums

Using the definition of semirings, can now generalize our definition of weights assigned to paths as:

\[
w(\pi) = \lambda(p(\pi)) \otimes \left( \bigotimes_{e \in \pi} w(e) \right) \otimes \rho(n(\pi))
\]

When we wish to aggregate weights over a set of paths \( P \) (for example, all the possible paths producing a string \( s \in \Sigma^* \)), we now have

\[
w(P) = \bigoplus_{\pi \in P} w(\pi)
\]

\[
= \bigoplus_{\pi \in P} \left( \lambda(p(\pi)) \otimes \left( \bigotimes_{e \in \pi} w(e) \right) \otimes \rho(n(\pi)) \right)
\]
3 Example question

Consider the following definitions:

\[ K = \mathbb{R}_{\geq 0} \times 2^Q \]

\[ (a, A) \otimes (b, B) = (a \times b, A \cup B) \]

\[ \mathbb{I} = (1, \emptyset) \]

\[ (a, A) \oplus (b, B) = \begin{cases} (a, A) & \text{if } a > b \\ (a, A \cup B) & \text{if } a = b \\ (b, B) & \text{if } a < b \end{cases} \]

\[ \mathbb{0} = (0, \emptyset) \]

Is this a semiring? \textbf{No.} For some \( a \in K \), \( a \otimes \mathbb{0} \neq \mathbb{0} \). How can this be fixed? Assume we have a WFSA with weights \( w: E \mapsto \mathbb{R}_{\geq 0} \), and starting and ending weights 1. Redefine the weight function such that \( w'(e) = (w(e), p(e)) \), what quantity does the path-sum correspond to?

4 Path expressions and computing the path-sum

Assume we have a semiring-weighted WFSA \( M = (Q, \Sigma, I, F, E, \lambda, \rho, w) \). For simplicity, further assume that \( I = \{q_0\} \), \( F = \{q_f\} \), \( \lambda(q_0) = \mathbb{I} \), and \( \rho(q_f) = \mathbb{I} \). Recall from the lectures on formal language theory that \textbf{regular expression} is a description of the languages accepted by an FSA, which can be defined inductively:

- \( \emptyset \) is a regular expression;
- \( \varepsilon \) is a regular expression;
- for each \( a \in \Sigma \), \( a \) is a regular expression denoting \( \{a\} \);
- if \( r \) and \( s \) are regular expressions denoting, respectively, the languages \( R \) and \( S \), then
  - \( (r + s) \) denotes \( R \cup S \);
  - \( (rs) \) denotes \( R \cdot S \); and
  - \( r^* \) denotes \( R^* \).

Recall that there is an algorithm for converting any deterministic FSA into a regular expression.

4.1 Path expressions

A path expression \( P(q, r) \) is a regular expression whose language \( L(P(q, r)) \) is the (possibly infinite) set of path strings \( \subseteq E^* \) leading from state \( q \in Q \) to state \( r \in Q \). We may construct this regular expression for any \( M \) by recalling that any deterministic FSA can be converted into an regular expression. Although \( M \) will not in general be deterministic, we will convert it into a \textbf{path automaton} \( M_\pi \) by setting \( \Sigma_\pi = E \) and letting each edge \( e \in E_\pi \) be labeled with the corresponding edge \( (q, \sigma, r) \in E \). This machine is trivially \textit{deterministic} since every edge has a unique label, and therefore it can be converted into a regular expression representing \( P(q, r) \).

\footnote{Any WFSA can be converted to this form by adding appropriately weighted \( \varepsilon \)-transitions.}
4.2 The path-sum algorithm

We are now in a position to define an algorithm that computes the $\oplus$-aggregation of all (possibly infinitely many!) paths in $P(q, r)$, that is:

$$w(P(q, r)) = \bigoplus_{\pi \in L(P(q, r))} |\pi| \prod_{i=1}^{\pi} w(e_i)$$

We define $w$ inductively as follows:

- $w(\emptyset) = 0$;
- $w(\varepsilon) = 1$;
- for each $e \in E (= \Sigma_\pi)$, $w(e)$ is defined according to $M$;
- if $r$ and $s$ are path expressions with weights $w(r)$ and $w(s)$, then
  - $w(r + s) = w(r) \oplus w(s)$;
  - $w(rs) = w(r) \otimes w(s)$; and
  - $w(r^*) = w(r)^*$.

Thus, computing the $\oplus$-aggregation of all paths in a machine $M$ can be accomplished by forming the path automaton $M_\pi$ and converting this into a regular expression.

5 The Forward algorithm (computes path-sum for acyclic WFSA)

The algorithm above is general, but when the WFSA does not contain any cycles or epsilons, we may define

**Require:** acyclic WFSA $M$ with a single start state $q_0$ and single end state $q_f$

```plaintext
for all $q \in Q - \{q_0\}$ do
d(q) ← 0
end for
d(q_0) ← \lambda(q_0)
perform a topological sort on $Q$
for all $q \in Q$ in top-sort order do
    for all $(q, x, r) \in E$ do
        $d(r) ← d(r) \oplus d(q) \otimes w(q, x, r)$
    end for
end for
return $d(q_f) \otimes \rho(q_f)$
```