

Measures on Languages

1 Introduction

1.1 Notation

Let Σ denote a non-empty, countable alphabet. Unless otherwise specified, assume $|\Sigma| < \infty$. Write ϵ to mean the empty string.

Let Σ^* be the set of all finite strings from Σ . Write strings as w or s , whichever happens to be more convenient.

Let L denote a language, implicitly over Σ . In other words, $L \subseteq \Sigma^*$.

We treat the empty language \emptyset as distinct from the language with a single empty string $\{\epsilon\}$.

Let \mathcal{L} be a family of languages.

Write deterministic finite automata shorthand as DFA, and non-deterministic finite automata shorthand as NFA.

Write regular expressions shorthand as regex.

If X is a set, then $\mathcal{P}(X)$ is the powerset of X .

Unless otherwise noted, vectors are implicitly in column format.

1.2 Regular Expressions

A regular expression over an alphabet Σ describes regular languages over Σ . Regular expressions are inductively generated, and we borrow heavily from Savage [2].

Definition 1 (Regular Expression). A regular expression over the finite alphabet Σ is defined inductively:

- (1) The empty language \emptyset is a regular expression.
- (2) The empty string ϵ is a regular expression denoting $\{\epsilon\}$.
- (3) For each $a \in \Sigma$, the standalone a is a regular expression denoting the singleton set $\{a\}$.
- (4) If r and s are regular expressions, then so are rs (string concat), $r + s$ (string choice), and r^* (string repeat).

Theorem 1 (Regular Expression Axioms). *Regular expressions satisfy the following axioms:*

- (1) $r\emptyset = \emptyset r = \emptyset$
- (2) $r\epsilon = \epsilon r = r$
- (3) $r + \emptyset = \emptyset + r = r$
- (4) $r + r = r$
- (5) $r + s = s + r$
- (6) $r(s + t) = rs + rt$
- (7) $(r + s)t = rt + st$
- (8) $r(st) = (rs)t$
- (9) $\emptyset^* = \epsilon$
- (10) $\epsilon^* = \epsilon$
- (11) $(\epsilon + r)^+ = r^*$
- (12) $(\epsilon + r)^* = r^*$
- (13) $r^*(\epsilon + r) = (\epsilon + r)r^* = r^*$
- (14) $r^*s + s = r^*s$
- (15) $r(sr)^* = (rs)^*r$
- (16) $(r + s)^* = (r^*s)^*r^* = (s^*r)^*s^*$

Outside of the Kleene star \star operation, axioms (1 - 8) effectively state that regular expressions are an idempotent semiring with additive constant \emptyset and multiplicative constant ϵ [1].

1.3 Representation of Regular Languages

There are several ways to represent regular languages, many of which can be found in literature [2]. We work with whatever is convenient for the problem at hand. For a regular language L over an alphabet Σ , there are a few notable ones:

- (1) **Sets:** sometimes if the language is finite or has a simple structure, a complete set presentation may be convenient.
- (2) **Regular expressions:** compact representation, also commonly used in practice when trying to do string matching.
- (3) **Finite state machines:** Savage [2] gives a fairly standard representation.

Definition 2 (DFA). A DFA A is a five-tuple $A = (\Sigma, Q, \delta, q_0, F)$, where Σ is the alphabet, Q is the finite set of states, $\delta: Q \times \Sigma \rightarrow Q$ is the transition function, q_0 is the initial state, and F is the set of final states.

For convenience, we might also write q_0 as q_1 , especially when talking about matrix indices. We'll try to remember to make note of when this rewriting is done.

Definition 3 (NFA). A NFA A is identically defined except for the transition function, which is now $\delta: Q \times \Sigma \rightarrow \mathcal{P}(Q)$. Each transition non-deterministically picks one state from the set.

- (4) **Matrices:** transition matrices can be constructed from both DFAs and NFAs. First, take $Q = \{q_1, q_2, \dots, q_n\}$. There are two primary possibilities:
 - (a) A matrix M_A corresponding to an automata A , with q_1 the initial state. Write $+ \{ \dots \}$ to denote the summation regular expression over a set. We construct the matrix as follows:

$$M_{A,i,j} = + \{ a : ((q_i, a), q_j) \in \delta \}$$

Here a is any character of Σ . In short, each entry of the matrix $M_{A,i,j}$ is the $+$ of all the characters that permit the transition from q_i to q_j .

If we take $v = ((1, 0, \dots))$ as an n -dimensional vector, where each coordinate i represents state q_i . Suppose that u is an n -dimensional vector indicating the final states, where $u_i = 1_{q_i \in F}$, then:

$$v^T M_A^k u$$

Will correspond to the regular expression of the sub-language of strings of precisely length k .

- (b) Alternatively we may see regular expressions as a set of matrices, each corresponding to a letter of Σ . In essence, for each $a \in \Sigma$, the associated matrix M_a has form: $M_{a,i,j} = 1_{((q_i, a), q_j) \in \delta}$, and indicates an adjacency transition matrix. The matrix described earlier can be recovered by observing that:

$$M_A = \sum_{a \in \Sigma} a M_a$$

2 Measures on Languages

Given a family of languages \mathcal{L} , let $\sigma(\mathcal{L})$ be the σ -algebra generated on \mathcal{L} satisfying the following:

(1)

$$\emptyset, \Sigma^* \in \sigma(\mathcal{L})$$

(2)

$$L \in \sigma(\mathcal{L}) \implies L^c = \Sigma^* \setminus L \in \sigma(\mathcal{L})$$

(3)

$$L_0, L_1, \dots \in \sigma(\mathcal{L}) \implies \bigcup_{k=0}^{\infty} L_k \in \sigma(\mathcal{L})$$

Then $(\mathcal{L}, \sigma(\mathcal{L}))$ is a measurable space.

Remark 1. If \mathcal{L} happened to be a family of regular languages, there is no guarantee that $\sigma(\mathcal{L})$ will still be a family of regular languages. A counter example is the following:

$$L_0 = \{\varepsilon\} \quad L_1 = \{ab\} \quad L_2 = \{aabb\} \quad \dots \quad L_k = \{a^k b^k\} \quad \dots$$

But taking the countable union yields:

$$\bigcup_{k=0}^{\infty} L_k = \{a^k b^k : k \in \mathbb{Z}_{\geq 0}\}$$

Which is not regular.

2.1 Measure 1: From Non-negative Integers

We first consider the non-negative integers $\mathbb{Z}_{\geq 0}$. Let η be a σ -finite measure on $\mathbb{Z}_{\geq 0}$. The σ -finite conditions ensures that no strange singularities occur for any integers under consideration. We may later restrict η to be finite if necessary, if we want nicer conditions.

Observe that, by abuse of notation:

$$\Sigma^* = \bigcup_{k=0}^{\infty} \Sigma^k$$

In English: Σ^* is the union of the set (language) of finite strings of length k , denoted Σ^k .

Because we assumed $|\Sigma| < \infty$, this also means that: $|\Sigma^k| = |\Sigma|^k$.

Consider now some language $L \in \sigma(\mathcal{L})$. Also decompose L into disjoint sub-languages by length as follows, with convenient subscripting:

$$L = \bigcup_{k=0}^{\infty} L_k$$

Of course, $L_k \subseteq \Sigma^k$.

Because we are able to precisely calculate $|\Sigma^k|$, one “natural” way of defining a measure λ_η on the measurable space $(\mathcal{L}, \sigma(\mathcal{L}))$ is as follows:

$$\lambda_\eta(L) = \sum_{k=0}^{\infty} \lambda_\eta(L_k) = \sum_{k=0}^{\infty} \frac{|L_k|}{|\Sigma^k|} \eta(k)$$

We claim that $(\mathcal{L}, \sigma(\mathcal{L}), \lambda_\eta)$ forms a measure space.

Theorem 2. λ_η is a measure.

Proof. We check (1) measure under empty set is zero and (2) countable additivity, which will satisfy the requirements of a measure.

- (1) Observe that $\lambda_\eta(\emptyset) = 0$ because the sum will be trivial.
- (2) Let L_0, L_1, L_2, \dots be a countable collection of pairwise disjoint languages. We decompose each of these languages into a countably indexed set, where $L_{j,k}$ is the j th language’s sub-language that only contains strings of length k . In other words:

$$L_j = \bigcup_{k=0}^{\infty} L_{j,k}$$

Observe that by (de)-construction, for any fixed j and for all $k_1 \neq k_2$, we have L_{j,k_1} and L_{j,k_2} are pairwise disjoint.

However, we have a stronger condition because each L_j is assumed to be pairwise

disjoint. Thus, for all $j_1 \neq j_2$ and $k_1 \neq k_2$, L_{j_1, k_1} and L_{j_2, k_2} are disjoint. Then:

$$\begin{aligned}
\lambda_\eta \left(\bigcup_{j=0}^{\infty} L_j \right) &= \lambda_\eta \left(\bigcup_{j=0}^{\infty} \bigcup_{k=0}^{\infty} L_{j,k} \right) \\
&= \sum_{j=0}^{\infty} \lambda_\eta \left(\bigcup_{k=0}^{\infty} L_{j,k} \right) \\
&= \sum_{j=0}^{\infty} \sum_{k=0}^{\infty} \frac{|L_{j,k}|}{|\Sigma^k|} \eta(k) \\
&= \sum_{j=0}^{\infty} \lambda_\eta(L_j)
\end{aligned}$$

This shows that λ_η is indeed a measure. □

2.2 Measure 2: Extending the Above

We may generalize λ_η as defined before slightly. Recall the definition, where L_0, L_1, L_2, \dots again defines a partition of L by size:

$$\lambda_\eta(L) = \sum_{k=0}^{\infty} \frac{|L_k|}{|\Sigma^k|} \eta(k)$$

Instead of dividing out by $|\Sigma^k|$ at each iteration of the sum, we may take a countable series of measures $\nu = \{\nu_0, \nu_1, \nu_2, \dots\}$, where each ν_k has support on precisely Σ^k . Then, define $\lambda_{\eta, \nu}$ as follows, taking again L_0, L_1, L_2, \dots the size partition of L :

$$\lambda_{\eta, \nu} = \sum_{k=0}^{\infty} \nu_k(L_k) \eta(k)$$

Often it's probably convenient to just assume each $\nu_k \in N$ to be the uniform distribution probability measure, which gets us λ_η as defined above.

Theorem 3. $\lambda_{\eta, \nu}$ is a measure.

Proof. We take a similar approach as before, and show (1) measure under empty set is zero and (2) countable additivity, which will show that $\lambda_{\eta, \nu}$ is indeed a measure.

- (1) Again, observe that $\lambda_{\eta, \nu}(\emptyset) = 0$ since the sum will be trivial.
- (2) Take L_0, L_1, L_2, \dots to be a countable collection of pairwise disjoint languages. Implicitly define a countably indexed set, where we take each $L_{j,k}$ as the j th language's sub-language with only strings of length k .

As with before, for $j_1 \neq j_2$ and $k_1 \neq k_2$, every L_{j_1, k_1} and L_{j_2, k_2} are pairwise disjoint. Then, doing the calculation:

$$\begin{aligned}
\lambda_{\eta, \nu} \left(\bigcup_{j=0}^{\infty} L_j \right) &= \lambda_{\eta, \nu} \left(\bigcup_{j=0}^{\infty} \bigcup_{k=0}^{\infty} L_{j, k} \right) \\
&= \sum_{j=0}^{\infty} \lambda_{\eta, \nu} \left(\bigcup_{k=0}^{\infty} L_{j, k} \right) \\
&= \sum_{j=0}^{\infty} \sum_{k=0}^{\infty} \nu_k(L_j) \eta(k) \\
&= \sum_{j=0}^{\infty} \lambda_{\eta, \nu}(L_j)
\end{aligned}$$

$\lambda_{\eta, \nu}$ is therefore a measure.

□

3 Approximating Languages

Given a measure space $(\mathcal{L}, \sigma(\mathcal{L}), \lambda)$, we may consider how similar two languages are. For two languages $L_1, L_2 \in \sigma(\mathcal{L})$, a “natural” difference is to consider their symmetric set difference:

$$d(L_1, L_2) = \lambda(L_1 \triangle L_2) = \lambda((L_1 \setminus L_2) \cup (L_2 \setminus L_1))$$

Recall that by the definition of a σ -algebra, the symmetric set difference $L_1 \triangle L_2$ is in $\sigma(\mathcal{L})$, and therefore measurable.

We hope restrict our attention to regular languages for now, or in other words, the class of languages precisely recognized by DFAs, and ask the following:

Question 1. *Given a regular language L recognized by a minimal DFA A and some $\varepsilon > 0$, does there exist a regular language L' recognized by A' such that A' has less states than A , and $d(L, L') < \varepsilon$?*

Example 1. Consider $\Sigma = \{a\}$, where $L_1 = aa^*$ and $L_2 = aaa^*$, and assume a geometric probability measure for η with success of probability $0 < p \leq 1$, through which λ is defined.

Recall that for the geometric distribution where $n \in \mathbb{Z}_+$:

$$\eta(n) = (1 - p)^{n-1} p$$

Observe that for L_1 and L_2 , we have:

$$L_1 = \underbrace{\{\}}_{\text{length 0}} \cup \underbrace{\{a\}}_{\text{length 1}} \cup \underbrace{\{aa\}}_{\text{length 2}} \cup \underbrace{\{aaa\}}_{\text{length 3}} \cup \dots$$

$$L_2 = \underbrace{\{\}}_{\text{length 0}} \cup \underbrace{\{\}}_{\text{length 1}} \cup \underbrace{\{aa\}}_{\text{length 2}} \cup \underbrace{\{aaa\}}_{\text{length 3}} \cup \dots$$

In other words, the only set on which the two languages differ is strings of length 1, which L_1 has, but L_2 does not. For the difference, this then means that:

$$d(L_1, L_2) = \lambda(L_1 \triangle L_2) = \lambda(\{a\}) = \frac{|\{a\}|}{|\Sigma^k|} \eta(1) = \frac{1}{1} p = p$$

In other words, with probability p , we can distinguish random strings generated from L_1 and L_2 , where the probability distribution is geometric, and over the length of the strings. Selection of the strings once length is fixed is irrelevant because each set corresponding to a length contains only one string.

Assume a (regular) language L given as an automata or regular expression, whichever may be convenient, and measure λ_n . A few challenges lie ahead:

Question 2. *How can we quickly measure $\lambda_\eta(L)$?*

Question 3. *How can we quickly generate a word of some length from L ?*

The word “quickly” here is key: many methods that can be thrown together to answer these questions are intrinsically exponential, so if we can introduce sub-exponential time techniques that would be pretty cool.

3.1 Counting

How many unique strings of length k does an automata have? The question is relatively straightforward for a DFA, and slightly more complicated for a NFA.

Theorem 4. *There exists a polynomial-time algorithm that counts the number of strings accepted by a DFA.*

Proof. Consider the matrix M representation of a DFA $A = (\Sigma, Q, \delta, q_1, F)$, which we claim can be conjured in polynomial time. We define M' where $M'_{i,j} = 1_{M_{i,j} \neq \emptyset}$. In other words, M' is the adjacency matrix corresponding to the directed graph described by A and M .

Let u be the vector where each element is such that $u_i = 1_{q_i \in F}$, then:

$$(1, 0, 0, \dots) (M')^k u$$

Will count the number of strings of length k . The algorithm runs in time polynomial to k the length and $n = |Q|$ the number of states.

□

References

- [1] Dexter Kozen. “A completeness theorem for Kleene algebras and the algebra of regular events”. In: *Infor. and Comput.* 110.2 (1994), pp. 366–390.
- [2] John E Savage. *Models of computation*. Vol. 136. Addison-Wesley Reading, MA, 1998.