Deterministic Finite Automata

Definition:

Σ is an alphabet
$\Sigma* = \{\alpha | \alpha = \beta\gamma\}$ is the set of strings over Σ
$(Q, \delta, \Sigma, q_0, F)$ is a DFA, where
$Q = \{q_0, q_1, ..., q_n\}$ is the set of states
Σ is an alphabet of symbols
q_0 is the start state
$F \in Q$ is the set of accepting states

Definition: $L(A)$ is the set of strings accepted by a DFA A.

Let $S^+ \subset \Sigma$ be a set of strings accepted by a DFA A.
Let S^- be the set of strings rejected by a DFA A.

Equivalence Relations for DFA's

Definition: A Myhill-Nerode equivalence on Σ^* for a DFA $A = (Q, \delta, \Sigma, q_0, F)$

Definition: Given a DFA D and $a, b \in \Sigma*$: $a \cong b$ i.f.f. $\delta*(q_0, a) = \delta*(q_0, b)$, and it is called an equivalence relation on $\Sigma*$

Definition: $[a]_{\cong} = \{b \in A | a \cong b\}$ is called an equivalence class of A.

Definition: A partition π of A is the set of equivalence classes where the union of the classes in π is equal to A.

A Quotient Automoton A_π can be constructed by partitioning the states of A into equivalence classes, and then by replacing each state that belongs in an equivalence relation with a new state that represents the merged states of that equivalence class

Definition: A Quotient Automaton is defined as $A_\pi = (Q_\pi, \Sigma, \delta_\pi, B(q_0, \pi), F_\pi$

$$
\begin{aligned}
&Q_\pi = \{B(q, \pi)| \ q \in Q\} \\
&F_\pi = \{B(q, \pi)| \ q \in F\} \\
&\delta_\pi : Q_\pi \to 2^{Q_\pi}
\end{aligned}
$$

$\forall a \in \Sigma, \ \forall B(q_i, \pi), \ B(q_j, \pi) \in Q_\pi$, The transition function is defined as

$$
\begin{aligned}
&\delta_\pi(B(q_i, \pi), a) = B(q_j, \pi), \\
&\quad \text{if } q_i, q_j \in Q \text{ and } \delta(q_i, a) = q_j
\end{aligned}
$$

A lattice is the set of all Quotient Automata obtained by merging the states of a DFA A.

The Learning Setup

We will say that $\{(x_i, y_i)\}$ is consistent with a DFA A if $\forall(x_i, y_i)$:

i) $x_i \in S^+$ i.f.f. $y_i = 1$
ii) $x_i \in S^-$ i.f.f. $y_i = -1$

Here is how the Parekh and Honavar define the learning environment:

- The sample space is defined as $X = \Sigma^*$
- A Concept $x \subset X$ is defined as $x = \{$x is a regular language $\}$
- The concept class C is the class of all DFA, where a concept x corresponds with some DFA in C.
- The target function is defined as $R : C \to \{-1, +1\}^*$

I find this setup of a learning space interesting for two reasons. First, R is a multiclass classifier. Second, while the concept is a regular language, the Concept Class is derived by mapping a concept to a DFA. This mapping is onto, but not one-to-one (**TODO:** explain this). However, if we were to restrict the mapping of regular languages to a canonical DFA, the relationship would be one-to-one.

Inside this environment, the training data D could be defined as:

$D = \{(x_i, y_i) : x_i \in \Sigma* \ and \ y_i \in \{-1, 1\}\}$, where $y_i = 1$ if $(x_i \in L(A))$

PAC Learnability

Leonard Pitt, in his paper 'Inductive Inference, DFAs, and Computational Complexity defines the PAC-Identifiability of DFas as follows:

DFAs are PAC-identifiable iff there exists a (possibly randomized algorithm A such that on input of any parameters ϵ and δ, for any DFA M of size n, for any number m, and for any probability distribution D on strings of $\Sigma*$ of length at most m, if A obtains labeled examples, M generated according to distribution D, then A produces a DFA M' such that, with probability at least $1 - \delta$, the probability (with respect to distribution D) of the set $\{w : w \in L(M) \oplus L(M')\}$ is at most ϵ. The run time of A (and hence the number of randomly generated examples obtained by A) is required to be polynomial in $n, m, \frac{1}{\epsilon}, \frac{1}{\delta}$, and $|\Sigma|$

So, $P[|L(M) \oplus L(M')| > \epsilon] \leq \frac{1}{\delta}$.

RPNI

Algorithm 1 RPNI

Input: $D = (\{x_i, y_i\})$
$\pi = \pi_0 = \{\{0\}, ..., \{\bar{N} - 1\}\}$
FOR $i = 1..\bar{N} - 1$
FOR $j = 0..i - 1$
$\tilde{\pi} = \pi \backslash \{\{B(i, \pi), B(j, \pi)\} \cup \{\{B(i, \pi), B(j, \pi)\}$
$M_{\tilde{\pi}} = derive(M, \pi)$
$\hat{\pi} = \ deterministicMerge(M, \tilde{\pi})$
if $consistent(M_{\hat{\pi}}), S^-)$
then
$M_\pi = M_{\hat{\pi}}$
$\pi = \hat{\pi}$
break
return M_π