Kernel Methods for Learning Languages

Leonid (Aryeh) Kontorovich [a] and Corinna Cortes [b] and Mehryar Mohri [c,b]

[a] *Department of Mathematics*
Weizmann Institute of Science, Rehovot, Israel 76100

[b] *Google Research,*
76 Ninth Avenue, New York, NY 10011

[c] *Courant Institute of Mathematical Sciences,*
251 Mercer Street, New York, NY 10012.

Abstract

This paper studies a novel paradigm for learning formal languages from positive and negative examples which consists of mapping strings to an appropriate high-dimensional feature space and learning a separating hyperplane in that space. Such mappings can often be represented flexibly with string kernels, with the additional benefit of computational efficiency. The paradigm inspected can thus be viewed as that of using kernel methods for learning languages.

We initiate the study of the linear separability of automata and languages by examining the rich class of piecewise-testable languages. We introduce a subsequence feature mapping to a Hilbert space and prove that piecewise-testable languages are linearly separable in that space. The proof makes use of word combinatorial results relating to subsequences. We also show that the positive definite symmetric kernel associated to this embedding is a rational kernel and show that it can be computed in quadratic time using general-purpose weighted automata algorithms. Our examination of the linear separability of piecewise-testable languages leads us to study the general problem of separability with other finite regular covers. We show that all languages linearly separable under a regular finite cover embedding, a generalization of the subsequence embedding we use, are regular.

We give a general analysis of the use of support vector machines in combination with kernels to determine a separating hyperplane for languages and study the corresponding learning guarantees. Our analysis includes several additional linear separability results in abstract settings and partial characterizations for the linear separability of the family of all regular languages.

Key words: finite automata, learning automata, margin theory, support vector machines, kernels, piecewise-testable languages.

1 Motivation

The problem of automatically learning a language from examples is among the most difficult problems of computer science and formal language theory. Most instances of this problem are provably hard, even in the specific case of learning finite automata or, equivalently, regular languages. This problem has been extensively studied over the last few decades.

On the negative side, the natural Occam learning attempt of finding the smallest automaton consistent with a set of accepted and rejected strings was shown to be NP-complete by Angluin [2] and Gold [14]. Pitt and Warmuth [28] further strengthened these results by showing that even an approximation within a polynomial function of the size of the smallest automaton is NP-hard. These results imply the computational intractability of the general problem of passively learning finite automata within many learning models, including the mistake bound model of Haussler et al. [16] or the PAC-learning model of Valiant [18].This last negative result can also be directly derived from the straightforward observation that the VC-dimension of finite automata is infinite.

On the positive side, Trakhtenbrot and Barzdin [31] showed that the smallest finite automaton consistent with the input data can be learned exactly from a uniform complete sample, whose size is exponential in the size of the automaton. The worst case complexity of their algorithm is exponential but a better average-case complexity can be obtained assuming that the topology and the labeling are selected randomly [31] or even that the topology is selected adversarially [11].

The model of identification in the limit of automata was introduced and discussed by Gold [13]. Deterministic finite automata were shown not to be identifiable in the limit from positive examples [13]. But positive results were given for the identification in the limit of the families of k-reversible languages [3] and subsequential transducers [26]. Some restricted classes of probabilistic automata such as acyclic probabilistic automata were also shown by Ron et al. to be efficiently learnable [29].

There is a vast literature dealing with the problem of learning automata and a comprehensive survey would be beyond the scope of this paper. Let us mention however that the algorithms suggested for learning automata are typically based on a state-merging idea. An initial automaton or prefix tree accepting the sample strings is first created. Then, starting with the trivial partition

Email addresses: aryehk@wisdom.weizmann.ac.il (Leonid (Aryeh) Kontorovich), corinna@google.com (Corinna Cortes), mohri@cims.nyu.edu (Mehryar Mohri).

with one state per equivalence class, classes are merged while preserving an invariant congruence property. The automaton learned is obtained by merging states according to the resulting classes. Thus, the choice of the congruence determines the algorithm.

This work departs from the established paradigm just described in that it does not use the state-merging technique. Instead, it initiates the study of linear separation of automata or languages by mapping strings to an appropriate high-dimensional feature space and learning a separating hyperplane in that space. Such mappings can be represented with much flexibility by string kernels, which can also be significantly more efficient to compute than a dot product in that space. Thus, our study can be viewed as that of using kernel methods for learning languages, starting with the rich class of *piecewise-testable languages*.

Piecewise-testable languages form an important family of regular languages. They have been extensively studied in formal language theory [23] starting with the work of Imre Simon [30]. A language L is said to be *n-piecewise-testable*, $n \in \mathbb{N}$, if whenever u and v have the same subsequences of length at most n and u is in L, then v is also in L. A language L is said to be *piecewise testable* if it is n-piecewise-testable for some $n \in \mathbb{N}$.

For a fixed n, n-piecewise-testable languages were shown to be identifiable in the limit by García and Ruiz [12]. The class of n-piecewise-testable languages is finite and thus has finite VC-dimension. To the best of our knowledge, there has been no learning result related to the full class of piecewise-testable languages.

This paper introduces an embedding of all strings in a high-dimensional feature space and proves that piecewise-testable languages are finitely linearly separable in that space, that is linearly separable with a finite-dimensional weight vector. The proof is non-trivial and makes use of deep word combinatorial results relating to subsequences. It also shows that the positive definite kernel associated to this embedding can be computed in quadratic time. Thus, the use of support vector machines [6,9,32] in combination with this kernel and the corresponding learning guarantees are examined. Since the VC-dimension of the class of piecewise-testable languages is infinite, it is not PAC-learnable and we cannot hope to derive PAC-style bounds for this learning scheme. But, the finite linear separability of piecewise-testable helps us derive weaker bounds based on the concept of the margin.

The linear separability proof is strong in the sense that the dimension of the weight vector associated with the separating hyperplane is finite. This is related to the fact that a *regular finite cover* is used for the separability of piecewise testable languages. This leads us to study the general problem of

separability with other finite regular covers. We prove that languages separated with such regular finite covers are necessarily regular.

The paper is organized as follows. Section 2 introduces some preliminary definitions and notation related to strings, automata, and piecewise-testable languages. Section 3 presents the proof of the finite linear separability of piecewise-testable languages using a subsequence feature mapping. Section 4 uses margin bounds to examine how the support vector machine algorithm combined with this subsequence feature mapping or, equivalently, a subsequence kernel, can be used to learn piecewise-testable languages. Most of the results of this section are general and hold for any finite linear separability with kernels. Section 5 examines the general problem of separability with regular finite covers and shows that all languages separated using such covers are regular. Section 6 shows that the subsequence kernel associated to the subsequence feature mapping is a rational kernel and that it is efficiently computable using general-purpose algorithms. Several additional linear separability results in abstract settings and partial characterizations are collected in Sections A and B of the Appendix.

2 Preliminaries

In all that follows, Σ represents a finite alphabet. The length of a string $x \in \Sigma^*$ over that alphabet is denoted by $|x|$ and the complement of a subset $L \subseteq \Sigma^*$ by $\overline{L} = \Sigma^* \setminus L$. For any string $x \in \Sigma^*$, we denote by $x[i]$ the ith symbol of x, $i \leq |x|$. More generally, we denote by $x[i{:}j]$, the substring of contiguous symbols of x starting at $x[i]$ and ending at $x[j]$.

A string x is a *subsequence* of $y \in \Sigma^*$ if x can be derived from y by erasing some of y's characters. We will write $x \sqsubseteq y$ to indicate that x is a subsequence of y. The relation $\sqsubseteq$ defines a partial order over Σ^*. For $x \in \Sigma^n$, the *shuffle ideal* of x is defined as the set of all strings containing x as a subsequence:

$$\text{Ш}(x) = \{u \in \Sigma^* : x \sqsubseteq u\} = \Sigma^* x[1]\Sigma^* \dots \Sigma^* x[n]\Sigma^*. \tag{1}$$

The definition of piecewise-testable languages was given in the previous section. An equivalent definition is the following: a language is *piecewise-testable* (PT for short) iff it is a finite Boolean combination of shuffle ideals [30].

We will often use the *subsequence feature mapping* $\phi : \Sigma^* \to \mathbb{R}^{\mathbb{N}}$ which associates to $x \in \Sigma^*$ a binary vector $\phi(x) = (y_u)_{u \in \Sigma^*}$ whose non-zero components

correspond to the subsequences of x:[1]

$$y_u = \begin{cases} 1 & \text{if } u \sqsubseteq x, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The computation of the kernel associated to ϕ is based on *weighted finite-state transducers*. A weighted finite-state transducer T over the field $(\mathbb{R}, \oplus, \otimes, 0, 1)$ is an 8-tuple $T = (\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$ where Σ is the finite input alphabet of the transducer; Δ is the finite output alphabet; Q is a finite set of states; $I \subseteq Q$ the set of initial states; $F \subseteq Q$ the set of final states; $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times \mathbb{R} \times Q$ a finite set of transitions each with a weight w; $\lambda : I \to \mathbb{R}$ the initial weight function; and $\rho : F \to \mathbb{R}$ the final weight function mapping F to $\mathbb{R}$.

For a path π in a transducer, we denote by $p[\pi]$ the origin state of that path, by $n[\pi]$ its destination state, and by $w[\pi]$ its weight obtained by multiplying the weights of its constituent transitions. We also denote by $P(I, x, y, F)$ the set of paths from the initial states I to the final states F.

A transducer T is *regulated* if the output weight associated by T to any pair of input-output string (x, y) by:

$$T(x, y) = \sum_{\pi \in P(I,x,y,F)} \lambda(p[\pi]) \cdot w[\pi] \cdot \rho[n[\pi]] \tag{3}$$

is well-defined and in $\mathbb{R}$. $T(x, y) = 0$ when $P(I, x, y, F) = \emptyset$. If for all $q \in Q$ $\sum_{\pi \in P(q,\epsilon,\epsilon,q)} w[\pi] \in \mathbb{R}$, then T is regulated. In particular, when T has no ϵ-cycle, it is regulated. The weighted transducers we will be considering in this paper will be regulated.

For any transducer T, we denote by T^{-1} its *inverse*, that is the transducer obtained from T by swapping the input and output label of each transition. The *composition* of two weighted transducers T_1 and T_2 with the same input and output alphabets Σ is a weighted transducer denoted by $T_1 \circ T_2$ when the sum:

$$(T_1 \circ T_2)(x, y) = \sum_{z \in \Sigma^*} T_1(x, z) \cdot T_2(z, y) \tag{4}$$

is well-defined and in $\mathbb{R}$ for all $x, y \in \Sigma^*$ [21].

3 Linear Separability of Piecewise-Testable Languages

This section shows that any piecewise-testable language is finitely linearly separable for the subsequence feature mapping.

[1] Elements $u \in \Sigma^*$ can be used as indices since Σ^* and $\mathbb{N}$ are isomorphic.

We will show that every piecewise-testable language is given by some *decision list* of shuffle ideals (a rather special kind of Boolean function). This suffices to prove the finite linear separability of piecewise-testable languages since decision lists are known to be linearly separable Boolean functions [4].

We will say that a string $u \in \Sigma^*$ is *decisive* for a language $L \subseteq \Sigma^*$, if $\text{Ш}(u) \subseteq L$ or $\text{Ш}(u) \subseteq \overline{L}$. The string u is said to be *positive-decisive* for L when $\text{Ш}(u) \subseteq L$ (*negative-decisive* when $\text{Ш}(u) \subseteq \overline{L}$). Note that when u is positive-decisive (negative-decisive),

$$x \in \text{Ш}(u) \Rightarrow x \in L \quad (\text{resp. } x \in \text{Ш}(u) \Rightarrow x \notin L). \tag{5}$$

Lemma 1 (Decisive strings) *Let $L \subseteq \Sigma^*$ be a piecewise-testable language, then there exists a decisive string $u \in \Sigma^*$ for L.*

Proof. We will prove that the existence of a decisive string is a property that holds for shuffle ideals and that it is preserved under the Boolean operations (negation, intersection, union). This will imply that it holds for all finite Boolean combinations of shuffle ideals, i.e., for all PT languages.

By definition, a shuffle ideal $\text{Ш}(u)$ admits u as a decisive string. It is also clear that if u is decisive for some PT language L, then u is also decisive for $\overline{L}$. Thus, the existence of a decisive string is preserved under negation. For the remainder of the proof, L_1 and L_2 will denote two PT languages over Σ.

If u_1 is positive-decisive for L_1 and u_2 is positive-decisive for L_2, $\text{Ш}(u_1) \cap \text{Ш}(u_2) \subseteq L = L_1 \cap L_2$. $\text{Ш}(u_1) \cap \text{Ш}(u_2)$ is not empty since it contains, for example, u_1u_2. For any string $u \in \text{Ш}(u_1) \cap \text{Ш}(u_2)$, $\text{Ш}(u) \subseteq \text{Ш}(u_1) \cap \text{Ш}(u_2)$, thus any such u is positive-decisive for L. Similarly, when u_1 is negative-decisive for L_1 and u_2 negative-decisive for L_2 any $u \in \text{Ш}(u_1) \cup \text{Ш}(u_2)$ is negative-decisive for $L = L_1 \cap L_2$. Finally, if u_1 is positive-decisive for L_1 and u_2 negative-decisive for L_2 then any $u \in \text{Ш}(u_2)$ is negative-decisive for $L = L_1 \cap L_2 \subseteq L_1$. This shows that the existence of a decisive string is preserved under intersection.

The existence of a decisive string is also preserved under union. If u_1 is positive-decisive for L_1 and u_2 positive-decisive for L_2, then any $u \in \text{Ш}(u_1) \cup \text{Ш}(u_2)$ is positive-decisive for $L = L_1 \cup L_2$. Similarly, when u_1 is negative-decisive for L_1 and u_2 negative-decisive for L_2, any $u \in \text{Ш}(u_1) \cap \text{Ш}(u_2) \neq \emptyset$ is negative-decisive for $L = L_1 \cup L_2$. Lastly, if u_1 is positive-decisive for L_1 and u_2 is negative-decisive for L_2 then any $u \in \text{Ш}(u_1)$ is positive-decisive for $L = L_1 \cup L_2$. □

We say that u is *minimally decisive* for L if it admits no proper subsequence

$v \sqsubseteq u$ that is decisive for L.

Lemma 2 (Finiteness of set of minimally-decisive strings) *Let $L \subseteq \Sigma^*$ be a PT language and let $D \subseteq \Sigma^*$ be the set of all minimally decisive strings for L, then D is a finite set.*

Proof. Observe that D is a *subsequence-free* subset of Σ^*: no element of D is a proper subsequence of another. Thus, the finiteness of D follows directly from Theorem 1 below. □

The following result, on which Lemma 2 is based, is a non-trivial theorem of word combinatorics which was originally discovered, in different forms, by Higman [17] in 1952 and Haines [15] in 1969. The interested reader could refer to [24, Theorem 2.6] for a modern presentation.

Theorem 1 ([15,17]) *Let Σ be a finite alphabet and $L \subseteq \Sigma^*$ a language containing no two distinct strings x and y such that $x \sqsubseteq y$. Then L is finite.*

The definitions and the results just presented can be generalized to decisiveness modulo a set V: we will say that a string u is *decisive modulo some $V \subseteq \Sigma^*$* if $V \cap \text{Ш}(u) \subseteq L$ or $V \cap \text{Ш}(u) \subseteq \overline{L}$. As before, we will refer to the two cases as *positive-* and *negative-decisiveness modulo V* and similarly define *minimally decisive strings modulo V*. These definitions coincide with ordinary decisiveness when $V = \Sigma^*$.

Lemma 3 (Finiteness of set of minimally-decisive strings modulo V) *Let $L, V \subseteq \Sigma^*$ be two PT languages and let $D \subseteq \Sigma^*$ be the set of all minimally decisive strings for L modulo V, then D is a non-empty finite set.*

Proof. Lemma 1 on the existence of decisive strings can be generalized straightforwardly to the case of decisiveness modulo a PT language V: if $L, V \subseteq \Sigma^*$ are PT and $V \neq \emptyset$, then there exists $u \in V$ such that u is decisive modulo V for L. Indeed, by Lemma 1, for any language of the form $\text{Ш}(s)$ there exists a decisive string $u \in V \cap \text{Ш}(s)$. The generalization follows by replacing $\text{Ш}(X)$ with $V \cap \text{Ш}(X)$ in the proof of Lemma 1.

Similarly, in view of Lemma 2, it is clear that there can only be finitely many minimally decisive strings for L modulo V. □

Theorem 2 (PT decision list) *If $L \subseteq \Sigma^*$ is PT then L is equivalent to some finite decision list Δ over shuffle ideals.*

Proof. Consider the sequence of PT languages $V_1, V_2, \ldots$ defined according to the following process:

- $V_1 = \Sigma^*$.
- When $V_i \neq \emptyset$, V_{i+1} is constructed from V_i in the following way. Let $D_i \subseteq V_i$ be the nonempty and finite set of minimally decisive strings u for L modulo V_i. The strings in D_i are either all positive-decisive modulo V_i or all negative-decisive modulo V_i. Indeed, if $u \in D_i$ is positive-decisive and $v \in D_i$ is negative-decisive then $uv \in Ш(u) \cap Ш(v)$, which generates a contradiction. Define σ_i as $\sigma_i = 1$ when all strings of D_i are positive-decisive, $\sigma_i = 0$ when they are negative-decisive modulo V_i and define V_{i+1} by:

$$V_{i+1} = V_i \setminus Ш(D_i), \tag{6}$$

  with $Ш(D_i) = \bigcup_{u \in D_i} Ш(u)$.

We show that this process terminates, that is $V_{N+1} = \emptyset$ for some $N > 0$. Assume the contrary. Then, the process generates an infinite sequence $D_1, D_2, \ldots$ Construct an infinite sequence $X = (x_n)_{n \in \mathbb{N}}$ by selecting a string $x_n \in D_n$ for any $n \in \mathbb{N}$. By construction, $D_{n+1} \subseteq \overline{Ш(D_n)}$ for all $n \in \mathbb{N}$, thus all strings x_n are necessarily distinct. Define a new sequence $(y_n)_{n \in \mathbb{N}}$ by: $y_1 = x_1$ and $y_{n+1} = x_{\xi(n)}$, where $\xi : \mathbb{N} \to \mathbb{N}$ is defined for all $n \in \mathbb{N}$ by:

$$\xi(n) = \begin{cases} \min\{k \in \mathbb{N} : \{y_1, \ldots, y_n, x_k\} \text{ is subsequence-free}\}, & \text{if such a } k \text{ exists,} \\ \infty & \text{otherwise.} \end{cases}$$

We cannot have $\xi(n) \neq \infty$ for all $n > 0$ since the set $Y = \{y_1, y_2, \ldots\}$ would then be (by construction) subsequence-free and infinite. Thus, $\xi(n) = \infty$ for some $n > 0$. But then any x_k, $k \in \mathbb{N}$, is a subsequence of an element of $\{y_1, \ldots, y_n\}$. Since the set of subsequences of $\{y_1, \ldots, y_n\}$ is finite, this would imply that X is finite and lead to a contradiction.

Thus, there exists an integer $N > 0$ such that $V_{N+1} = \emptyset$ and the process described generates a finite sequence $D = (D_1, \ldots, D_N)$ of nonempty sets as well as a sequence $\sigma = (\sigma_i) \in \{0,1\}^N$. Let Δ be the decision list

$$(Ш(D_1), \sigma_1), \ldots, (Ш(D_N), \sigma_N). \tag{7}$$

Let $\Delta_n : \Sigma^* \to \{0,1\}$, $n = 1, \ldots, N$, be the mapping defined for all $x \in \Sigma^*$ by:

$$\forall x \in \Sigma^*, \quad \Delta_n(x) = \begin{cases} \sigma_n & \text{if } x \in Ш(D_n), \\ \Delta_{n+1}(x) & \text{otherwise,} \end{cases} \tag{8}$$

with $\Delta_{N+1}(x) = \sigma_N$. It is straightforward to verify that Δ_n coincides with the characteristic function of L over $\bigcup_{i=1}^{n} Ш(D_i)$. This follows directly from the

definition of decisiveness. In particular, since

$$V_n = \bigcap_{i=1}^{n-1} \overline{\text{Ш}(D_i)} \tag{9}$$

and $V_{N+1} = \emptyset$,

$$\bigcup_{i=1}^{N} \text{Ш}(D_i) = \Sigma^*, \tag{10}$$

and Δ coincides with the characteristic function of L everywhere. □

Using this result, we show that a PT language is linearly separable with a finite-dimensional weight vector.

Corollary 1 *For any PT language L, there exists a weight vector $w \in \mathbb{R}^{\mathbb{N}}$ with finite support such that $L = \{x : \langle w, \phi(x)\rangle > 0\}$, where ϕ is the subsequence feature mapping.*

Proof. Let L be a PT language. By Theorem 2, there exists a decision list $(\text{Ш}(D_1), \sigma_1), \ldots, (\text{Ш}(D_N), \sigma_N)$ equivalent to L where each D_n, $n = 1, \ldots, N$, is a finite set. We construct a weight vector $w = (w_u)_{u\in\Sigma^*} \in \mathbb{R}^{\mathbb{N}}$ by starting with $w = 0$ and modifying its coordinates as follows in the order $n = N, N-1, \ldots, 1$:

$$\forall u \in D_n, \quad w_u = \begin{cases} +\Big(|\sum_{v\in V^-} w_v| + 1\Big) & \text{if } \sigma_n = 1, \\ -\Big(|\sum_{v\in V^+} w_v| + 1\Big) & \text{otherwise,} \end{cases} \tag{11}$$

where V^- and V^+ denote

$$V^- = \{v \in \bigcup_{i=n+1}^{N} D_i : w_v < 0\} \quad \text{and} \quad V^+ = \{v \in \bigcup_{i=n+1}^{N} D_i : w_v > 0\}. \tag{12}$$

By construction, the decision list is equivalent to $\{x : \langle w, \phi(x)\rangle > 0\}$. Since each D_n, $n = 1, \ldots, N$, is finite, the weight vector w has only a finite number of non-zero coordinates. □

In particular, we obtain a new characterization of piecewise testability: a language is PT if and only if it is finitely linearly separable under the subsequence embedding. The "only if" direction is entailed by Corollary 1, while the "if" direction is a consequence of Theorem 5, proved below.

The dimension of the feature space associated to ϕ is infinite. Section 6 will show however that the kernel associated to ϕ can be computed efficiently. Lin-

ear separability combined with the use of this kernel ensures efficient learnability, as we shall see in the next section.

4 Learning Linearly Separable Languages

This section deals with the problem of learning PT languages, and other linearly separable concept classes.

In the previous section, we showed that using the subsequence feature mapping ϕ, or equivalently the corresponding subsequence kernel K, PT languages are finitely linearly separable. In Section 6, we will show that $K(x, y)$ can be computed in $O(|\Sigma||x||y|)$ for any two strings $x, y \in \Sigma^*$. These results suggest the use of a linear separator learning technique such as support vector machines (SVMs) [6,9,32] combined with the subsequence kernel K for learning PT languages. In view of the complexity of the subsequence kernel computation just mentioned, the complexity of computing the SVM solution for a sample of size m with longest string $x_{\max}$ is $O(\mathrm{QP}(m)) + m^2 |x_{\max}|^2 |\Sigma|)$, where $\mathrm{QP}(m)$ is the cost of solving a quadratic programming problem of size m, which is at most $O(m^3)$.

We will use the standard margin bound to analyze the behavior of that algorithm. Note however, that since the VC-dimension of the set of PT languages is infinite, PAC-learning is not possible and we need to resort to a weaker guarantee.

Let $(x_1, y_1), \ldots, (x_m, y_m) \in X \times \{-1, +1\}$ be a labeled sample from a set X ($X = \Sigma^*$ when learning languages). The margin ρ of a hyperplane with weight vector $w \in \mathbb{R}^{\mathbb{N}}$ over this sample is defined by:

$$\rho = \inf_{i=1,\ldots,m} \frac{y_i \langle w, \phi(x_i) \rangle}{\|w\|}. \tag{13}$$

The sample is linearly separated by w iff $\rho > 0$. Note that our definition holds even for infinite-size samples.

The linear separation result shown for the class of PT languages is strong in the following sense. For any weight vector $w \in \mathbb{R}^{\mathbb{N}}$, let $\mathrm{supp}(w) = \{i : w_i \neq 0\}$ denote the support of w, then the following property holds for PT languages.

Definition 1 *Let C be a concept class defined over a set X; that is, $C \subseteq 2^X$. We will say that a concept $c \in C$ is* finitely linearly separable*, if there exists a mapping $\phi : X \to \{0, 1\}^{\mathbb{N}}$ and a weight vector $w \in \mathbb{R}^{\mathbb{N}}$ with* finite support*,*

$|\operatorname{supp}(w)| < \infty$, *such that*

$$c = \{x \in X : \langle w, \phi(x) \rangle > 0\}. \tag{14}$$

The concept class C is said to be finitely linearly separable *if all $c \in C$ are finitely linearly separable for the same mapping ϕ.*

Note that in general a linear separation in an infinite-dimensional space does not guarantee a strictly positive margin ρ. Points in an infinite-dimensional space may be arbitrarily close to the separating hyperplane and their infimum distance could be zero. However, finitely linear separation does guarantee a strictly positive margin.

Proposition 1 *Let C be a concept class defined over a set X that is finitely linearly separable using the mapping $\phi : X \to \{0,1\}^{\mathbb{N}}$ and a weight vector $w \in \mathbb{R}^{\mathbb{N}}$. Then, the margin ρ of the hyperplane defined by w is strictly positive, $\rho > 0$.*

Proof. By assumption, the support of w is finite. For any $x \in X$, let $\phi'(x)$ be the projection of $\phi(x)$ on the span of w, $\operatorname{span}(w)$. Thus, $\phi'(x)$ is a finite-dimensional vector for any $x \in X$ with discrete coordinates in $\{0,1\}$. Thus, the set of $S = \{\phi'(x) : x \in X\}$ is finite. Since for any $x \in X$, $\langle w, \phi(x) \rangle = \langle w, \phi'(x) \rangle$, the margin is defined over a finite set:

$$\rho = \inf_{x \in X} \frac{y_x \langle w, \phi'(x) \rangle}{\|w\|} = \min_{z \in S} \frac{y_x \langle w, z \rangle}{\|w\|} > 0, \tag{15}$$

and is thus strictly positive. □

By Corollary 1, PT languages are finitely linearly separable under the subsequence embedding. Thus, there exists a hyperplane separating a PT language with a strictly positive margin.

The following general margin bound holds for all classifiers consistent with the training data [5].

Theorem 3 (Margin bound) *Define the class $\mathcal{F}$ of real-valued functions on the ball of radius R in $\mathbb{R}^n$ as*

$$\mathcal{F} = \{x \mapsto \langle w, x \rangle : \|w\| \leq 1, \|x\| \leq R\}. \tag{16}$$

There is a constant α_0 such that, for all distributions D over X, with probability at least $1 - \delta$ over m independently generated examples, if a classifier $\operatorname{sgn}(f)$, with $f \in \mathcal{F}$, has margin at least ρ on the training examples, then the

generalization error of $\operatorname{sgn}(f)$ *is no more than*

$$\frac{\alpha_0}{m}\left(\frac{R^2}{\rho^2}\log^2 m + \log(\frac{1}{\delta})\right). \tag{17}$$

In general, linear separability does not provide a margin-based guarantee when the support of the weight vector is unbounded. Any sample of size m can be trivially made linearly separable by using an embedding $\phi : X \to \{0,1\}^{\mathbb{N}}$ mapping each point x to a distinct dimension. The margin ρ for such a mapping is $\frac{1}{2\sqrt{m}}$ and thus goes to zero as m increases, and the ratio $(R/\rho)^2$, where $R = 1$ is the radius of the sphere containing the sample points, is $(R/\rho)^2 = 4m$. The bound of Theorem 3 is not effective with that value of $(R/\rho)^2$. The following result shows however that linear separability with a *finite support weight vector* ensures a strictly positive margin and thus convergence guarantees.

Theorem 4 *Let C be a finitely linearly separable concept class over X with a feature mapping $\phi : X \to \{0,1\}^{\mathbb{N}}$. Define $\mathcal{F}$ as the class of real-valued functions*

$$\mathcal{F} = \{x \mapsto \langle w, \phi(x)\rangle : \|w\| \leq 1, \|\phi(x)\| \leq R\}. \tag{18}$$

There is a constant α_0 such that, for all distributions D over X, for any concept $c \in C$, there exists $\rho_0 > 0$ such that with probability at least $1-\delta$ over m independently generated examples according to D, there exists a classifier $\operatorname{sgn}(f)$*, with $f \in \mathcal{F}$, with margin at least ρ_0 on the training examples, and generalization error no more than*

$$\frac{\alpha_0}{m}\left(\frac{R^2}{\rho_0^2}\log^2 m + \log(\frac{1}{\delta})\right). \tag{19}$$

Proof. Fix a concept $c \in C$. By assumption, c is finitely linearly separable from $X \setminus c$ by some hyperplane. By Proposition 1, the corresponding margin ρ_0 is strictly positive, $\rho_0 > 0$. ρ_0 is less than or equal to the margin of the optimal hyperplane ρ separating c from $X \setminus c$ based on the m examples.

Since the full sample X is linearly separable, so is any subsample of size m. Let $f \in \mathcal{F}$ be the linear function corresponding to the optimal hyperplane over a sample of size m drawn according to D. Then, the margin of f is at least as large as ρ since not all points of X are used to define f. Thus, the margin of f is greater than or equal to ρ_0 and the statement follows Theorem 3. □

Theorem 4 applies directly to the case of PT languages since by Corollary 1 they are finitely linearly separable under the subsequence embedding. Observe that in the statement of the theorem, ρ_0 depends on the particular concept c learned but does not depend on the sample size m.

Note that the linear separating hyperplane with finite-support weight vector is not necessarily an optimal hyperplane. The following proposition shows however that this property holds when the mapping ϕ is surjective.

Proposition 2 *Let $c \in C$ be a finitely linearly separable concept with the feature mapping $\phi : X \to \{0,1\}^{\mathbb{N}}$ and weight vector w with finite support, $|\operatorname{supp}(w)| < \infty$, such that $\phi(X) = \mathbb{R}^{\mathbb{N}}$. Assume that ϕ is surjective, then the weight vector $\hat{w}$ corresponding to the optimal hyperplane for c has also a finite support and $\operatorname{supp}(\hat{w}) \subseteq \operatorname{supp}(w)$.*

Proof. Assume that $\hat{w}_i \neq 0$ for some $i \notin \operatorname{supp}(w)$. We first show that this implies the existence of two points $x_- \notin c$ and $x_+ \in c$ such that $\phi(x_-)$ and $\phi(x_+)$ differ only by their ith coordinate.

Let ϕ' be the mapping such that for all $x \in X$, $\phi'(x)$ differs from $\phi(x)$ only by the ith coordinate and let $\hat{w}'$ be the vector derived from $\hat{w}$ by setting the ith coordinate to zero. Since ϕ is surjective, thus $\phi^{-1}(\phi'(x)) \neq \emptyset$. If x and any $x' \in \phi^{-1}(\phi'(x))$ are in the same class for all $x \in X$, then

$$\operatorname{sgn}(\langle \hat{w}, \phi(x) \rangle) = \operatorname{sgn}(\langle \hat{w}, \phi'(x) \rangle). \tag{20}$$

Fix $x \in X$. Assume for example that $[\phi'(x)]_i = 0$ and $[\phi(x)]_i = 1$, then $\langle \hat{w}, \phi'(x) \rangle = \langle \hat{w}', \phi(x) \rangle$. Thus, in view of Equation 20,

$$\operatorname{sgn}(\langle \hat{w}, \phi(x) \rangle) = \operatorname{sgn}(\langle \hat{w}, \phi'(x) \rangle) = \operatorname{sgn}(\langle \hat{w}', \phi(x) \rangle). \tag{21}$$

We obtain similarly that $\operatorname{sgn}(\langle \hat{w}, \phi(x) \rangle) = \operatorname{sgn}(\langle \hat{w}', \phi(x) \rangle)$ when $[\phi'(x)]_i = 1$ and $[\phi(x)]_i = 0$. Thus, for all $x \in X$, $\operatorname{sgn}(\langle \hat{w}, \phi(x) \rangle) = \operatorname{sgn}(\langle \hat{w}', \phi(x) \rangle)$. This leads to a contradiction, since the norm of the weight vector for the optimal hyperplane is the smallest among all weight vectors of separating hyperplanes.

Since any pair x, x' as defined above cannot be in the same class, this proves the existence of $x_- \notin c$ and $x_+ \in c$ with $\phi(x_-)$ and $\phi(x_+)$ differing only by their ith coordinate.

But, since $i \notin \operatorname{supp}(w)$, for two such points $x_- \notin c$ and $x_+ \in c$, $\langle w, \phi(x_-) \rangle = \langle w, \phi(x_+) \rangle$. This contradicts the status of $\operatorname{sgn}(\langle w, \phi(x) \rangle)$ as a linear separator. Thus, our original hypothesis cannot hold: there exists no $i \notin \operatorname{supp}(w)$ such that $\hat{w}_i \neq 0$ and the support of $\hat{w}$ is included in that of w. □

In the following, we will give another analysis of the generalization error of SVMs for finitely separable hyperplanes using the bound of Vapnik based on

the number of essential support vectors: [2]

$$\mathrm{E}[error(h_m)] \leq \frac{\mathrm{E}[(\frac{R_{m+1}}{\rho_{m+1}})^2]}{m+1}, \tag{22}$$

where h_m is the optimal hyperplane hypothesis based on a sample of m points, $error(h_m)$ the generalization error of that hypothesis, R_{m+1} the smallest radius of a set of essential support vectors of an optimal hyperplane defined over a set of $m+1$ points, and ρ_{m+1} its margin.

Let c be a finitely separable concept. When the mapping ϕ is surjective, by Proposition 2, the weight vector $\hat{w}$ of the optimal separating hyperplane for c has finite support and the margin ρ_0 is positive, $\rho_0 > 0$. Thus, the smallest radius of a set of essential support vectors for that hyperplane is $R = \sqrt{N(c)}$ where $N(c) = |\operatorname{supp}(\hat{w})|$. If R_{m+1} tends to R when m tends to infinity, then for all $\epsilon > 0$, there exists M_ϵ such that for $m > M_\epsilon$, $R^2(m) \leq N(c) + \epsilon$. In view of Equation 22, the expectation of the generalization error of the optimal hyperplane based on a sample of size m is bounded by

$$\mathrm{E}[error(h_m)] \leq \frac{\mathrm{E}[(\frac{R_{m+1}}{\rho_{m+1}})^2]}{m+1} \leq \frac{N(c)+\epsilon}{\rho_0^2(m+1)}. \tag{23}$$

This upper bound varies as $\frac{1}{m}$.

5 Finite Cover with Regular Languages

In the previous sections, we introduced a feature mapping ϕ, the subsequence mapping, for which PT languages are finitely linearly separable. The subsequence mapping can be defined in terms of the set of shuffle ideals of all strings, $U_u = \text{Ш}(u)$, $u \in \Sigma^*$. A string x can belong only to a finite number of shuffle ideals U_u, which determine the non-zero coordinates of $\phi(x)$. This leads us to consider other such mappings based on other regular sets U_u and investigate the properties of languages linearly separated under such mappings. The main result of this section is that all such linearly separated languages are regular.

[2] A support vector $\phi(x)$, $x \in X$, is *essential* if $\phi(x) \in \mathrm{SV}(S)$ whenever $x \in S$, where $\mathrm{SV}(S)$ are the support vectors induced by the sample S.

5.1 Definitions

Let $U_n \subseteq \Sigma^*$, $n \in \mathbb{N}$, be a countable family of sets such that any string $x \in \Sigma^*$ lies in at most finitely many U_n. Thus, for all $x \in \Sigma^*$,

$$\sum_n \psi_n(x) < \infty, \tag{24}$$

where ψ_n is the characteristic function of U_n:

$$\psi_n(x) = \begin{cases} 1 & \text{if } x \in U_n \\ 0 & \text{otherwise.} \end{cases} \tag{25}$$

Any such family $(U_n)_{n\in\mathbb{N}}$ is called a *(locally) finite cover* of Σ^*. If additionally, each U_n is a regular set and Σ^* is a member of the family, we will say that $(U_n)_{n\in\mathbb{N}}$ is a *regular finite cover* (RFC).

Any finite cover $(U_n)_{n\in\mathbb{N}}$ naturally defines a positive definite symmetric kernel K over Σ^* given by:

$$\forall x, y \in \Sigma^*, \quad K(x, y) = \sum_n \psi_n(x)\psi_n(y). \tag{26}$$

Its finiteness, symmetry, and positive definiteness follow its construction as a dot product. $K(x, y)$ counts the number of common sets U_n that x and y belong to.

We may view $\psi(x)$ as an infinite-dimensional vector in the space $\mathbb{R}^{\mathbb{N}}$, in which case we can write $K(x, y) = \langle \psi(x), \psi(y) \rangle$. We will say that ψ is an *RFC-induced embedding*. Any weight vector $w \in \mathbb{R}^{\mathbb{N}}$ defines a language $L(w)$ given by:

$$L(w) = \{x \in \Sigma^* : \langle w, \psi(x) \rangle > 0\}. \tag{27}$$

Note that since Σ^* is a member of every RFC, $K(x, y) \geq 1$.

5.2 Main Result

The main result of this section is that any finitely linearly separable language under an RFC embedding is regular. The converse is clearly false. For a given RFC, not all regular languages can be defined by some separating hyperplane. A simple counterexample is provided with the RFC $\{\emptyset, U, \Sigma^* \setminus U, \Sigma^*\}$ where U is some regular language. For this RFC, U, its complement, Σ^*, and the empty set are linearly separable but no other regular language is.

Theorem 5 *Let $\psi : \Sigma^* \to \{0, 1\}^{\mathbb{N}}$ be an RFC-induced embedding and let*

$w \in \mathbb{R}^{\mathbb{N}}$ *be a finitely supported weight vector. Then, the language* $L(w) = \{x \in \Sigma^* : \langle w, \psi(x)\rangle > 0\}$ *is regular.*

Proof. Let $f : \Sigma^* \to \mathbb{R}$ be the function defined by:

$$f(x) = \langle w, \psi(x)\rangle = \sum_{i=1}^{N} w_i \psi_i(x), \tag{28}$$

where the weights $w_i \in \mathbb{R}$ and the integer $N = |\operatorname{supp}(w)|$ are independent of x. Observe that f can only take on finitely many real values $\{r_k : k = 1, \ldots, K\}$. Let $L_{r_k} \subseteq \Sigma^*$ be defined by

$$L_{r_k} = f^{-1}(r_k). \tag{29}$$

A subset $I \subseteq \{1, 2, \ldots, N\}$ is said to be r_k-*acceptable* if $\sum_{i\in I} w_i = r_k$. Any such r_k-acceptable set corresponds to a set of strings $L_I \subseteq \Sigma^*$ such that

$$L_I = \left(\bigcap_{i\in I} \psi_i^{-1}(1)\right) \setminus \left(\bigcup_{i\in\{1,\ldots,N\}\setminus I} \psi_i^{-1}(1)\right) = \left(\bigcap_{i\in I} U_i\right) \setminus \left(\bigcup_{i\in\{1,\ldots,N\}\setminus I} U_i\right).$$

Thus, L_I is regular because each U_i is regular by definition of the RFC. Each L_{r_k} is the union of finitely many r_k-acceptable L_I's, and L is the union of the L_{r_k} for positive r_k. □

Theorem 5 provides a representation of regular languages in terms of some subsets of $\mathbb{R}^{\mathbb{N}}$. Although we present a construction for converting this representation to a more familiar one such as a finite automaton, our construction is not necessarily efficient. Indeed, for some r_k there may be exponentially many r_k-acceptable L_Is. This underscores the specific feature of our method. Our objective is to learn regular languages efficiently using some representation, not necessarily automata.

5.3 Representer Theorem

Let $S = \{x_j : j = 1, \ldots, m\} \subseteq \Sigma^*$ be a finite set of strings and $\alpha \in \mathbb{R}^m$. The pair (S, α) defines a language $L(S, \alpha)$ given by:

$$L(S, \alpha) = \{x \in \Sigma^* : \sum_{j=1}^{m} \alpha_j K(x, x_j) > 0\}. \tag{30}$$

Let $w = \sum_{j=1}^{m} \alpha_j \psi(x_j)$. Since each $\psi(x_j)$ has only a finite number of non-zero components, the support of w is finite and by Theorem 5, $L(S, \alpha)$ can be seen to be regular. Conversely, the following result holds.

Theorem 6 *Let $\psi : \Sigma^* \to \{0,1\}^{\mathbb{N}}$ be an RFC-induced embedding and let $w \in \mathbb{R}^{\mathbb{N}}$ be a finitely supported weight vector. Let $L(w)$ be defined by $L(w) = \{x \in \Sigma^* : \langle w, \psi(x)\rangle > 0\}$. Then, there exist (x_j), $j = 1, \ldots, m$, and $\alpha \in \mathbb{R}^m$ such that $L(w) = L(S, \alpha) = \{x \in \Sigma^* : \sum_{j=1}^m \alpha_j K(x, x_j) > 0\}$.*

Proof. Without loss of generality, we can assume that no cover set $U_n \neq \Sigma^*$, U_n is fully contained in a finite union of the other cover sets $U_{n'}$, $U_{n'} \neq \Sigma^*$. Otherwise, the corresponding feature component can be omitted for linear separation. Now, for any $U_n \neq \Sigma^*$, let $x_n \in U_n$ be a string that does not belong to any finite union of $U_{n'}$, $U_{n'} \neq \Sigma^*$. For $U_n = \Sigma^*$, choose an arbitrary string $x_n \in \Sigma^*$. Then, by definition of the x_n,

$$\langle w, \psi(x)\rangle = \sum_{j=1}^{m} w_j K(x, x_j). \tag{31}$$

This proves the claim. □

This result shows that any finitely linearly separable language can be inferred from a finite sample.

5.4 Further Characterization

It is natural to ask what property of finitely supported hyperplanes is responsible for their inducing regular languages. In fact, Theorem 5 is readily generalized:

Theorem 7 *Let $f : \Sigma^* \to \mathbb{R}$ be a function such that there exist an integer $N \in \mathbb{N}$ and a function $g : \{0,1\}^N \to \mathbb{R}$ such that*

$$\forall x \in \Sigma^*, \quad f(x) = g(\psi_1(x), \psi_2(x), \ldots, \psi_N(x)), \tag{32}$$

Thus, the value of f depends on a fixed finite number of components of ψ. Then, for any $r \in \mathbb{R}$, the language $L = \{x \in \Sigma^* : f(x) = r\}$ is regular.

Proof. Since f is a function of finitely many binary variables, its range is finite. From here, the proof proceeds exactly as in the proof of Theorem 5, with identical definitions for $\{r_k\}$ and L_{r_k}. □

This leads to the following corollary.

Corollary 2 *Let $f : \Sigma^* \to \mathbb{R}$ be a function satisfying the conditions of Theorem 7. Then, for any $r \in \mathbb{R}$, the languages $L_1 = \{x \in \Sigma^* : f(x) > r\}$ and $L_2 = \{x \in \Sigma^* : f(x) < r\}$ are regular.*

6 Efficient Kernel Computation

The positive definite symmetric kernel K associated to the subsequence feature mapping ϕ is defined by:

$$\forall x, y \in \Sigma^*, \quad K(x, y) = \langle \phi(x), \phi(y) \rangle = \sum_{u \in \Sigma^*} [\![u \sqsubseteq x]\!] [\![u \sqsubseteq y]\!]. \tag{33}$$

where $[\![P]\!]$ represents the 0-1 truth value of the predicate P. Thus, $K(x, y)$ counts the number of subsequences common to x and y, without multiplicity.

This subsequence kernel is closely related to but distinct from the one defined by Lodhi et al. [22]. Indeed, the kernel of Lodhi et al. counts the number of occurrences of subsequences common to x and y. Thus, for example $K(abc, acbc) = 8$, since the cardinal of the set of common subsequences of abc and $acbc$, $\{\epsilon, a, b, c, ab, ac, bc, abc\}$, is 8. But, the kernel of Lodhi et al. (without penalty factor) would instead associate the value 10 to the pair $(abc, acbc)$, since each of c and ac occurs twice in the second string.

A string with n distinct symbols has at least 2^n possible subsequences, so a naive computation of $K(x, y)$ based on the enumeration of the subsequences of x and y is inefficient. We will show however that K is a positive definite symmetric rational kernel and that $K(x, y)$ can be computed in quadratic time, $O(|\Sigma||x||y|)$, using the general algorithm for the computation of rational kernels [7]. [3]

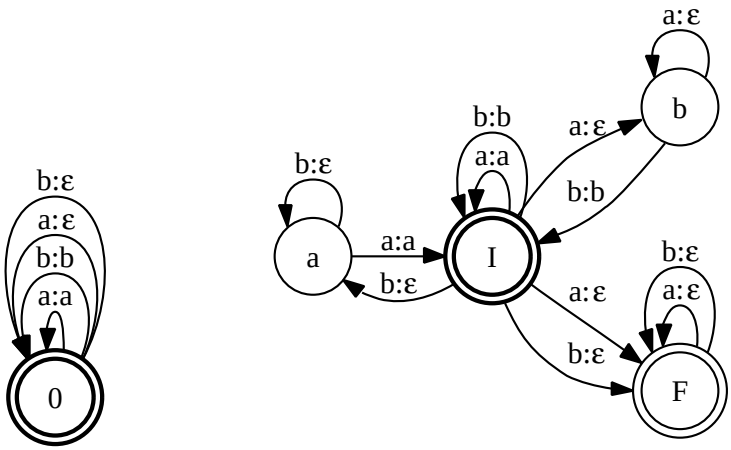
To do so, we will show that there exists a weighted transducer T over the semiring $(\mathbb{R}, +, \cdot, 0, 1)$ such that for all $x, y \in \Sigma^*$

$$K(x, y) = (T \circ T^{-1})(x, y). \tag{34}$$

This will prove that K is a rational kernel since it can be represented by the weighted transducer S and by a theorem of [7], it is positive definite symmetric since S has the form $S = T \circ T^{-1}$.

There exists a simple (unweighted) transducer T_0 mapping each string to the

[3] In previous work [20], we described a special-purpose method suggested by Derryberry [10] for computing K, which turns out to be somewhat similar to that of Lodhi et al.



(a) (b)

Fig. 1. Subsequence transducers for $\Sigma = \{a, b\}$. A bold circle indicates an initial state. Final states are marked with double-circles. (a) Transducer T_0 associating to each input string $x \in \Sigma^*$ the set of its subsequences with multiplicity. (b) Subsequence transducer T associating to each string $x \in \Sigma^*$ the set of its subsequences with multiplicity one even if the number of occurrences is high.

set of its subsequences defined by the following regular expression over pairs:

$$\bigcup_{a \in \Sigma} (a, a) \cup (a, \epsilon). \tag{35}$$

This is clear since, by definition, each symbol can be either left unchanged in the output, or replaced by the empty string ϵ, thereby generating all possible subsequences. Figure 1(a) shows that transducer in the particular case of an alphabet with just two symbols a and b. The transducer has only one state.

The transducer T_0 may generate several copies of the same subsequence of a sequence x. For example, the subsequence a of $x = aa$ can be generated by T_0 by either erasing the first symbol or the last symbol. To be consistent with the definition of the subsequence kernel K, we need instead to generate only one copy of each subsequence of a sequence. We will construct a transducer T that will do just that. To simplify the discussion, we will assume that the alphabet is reduced to $\Sigma = \{a, b\}$. The analysis extends to the general case straightforwardly.

T is constructed by removing some paths of T_0 to generate only the occurrence of a subsequence u of x whose symbols are read as early as possible. We can remove from T_0 paths containing a pattern described by $(b, \epsilon)(a, \epsilon)^*(b, b)$. That is because that subsequence can also be generated via $(b, b)(a, \epsilon)^*(b, \epsilon)$, which corresponds to an earlier instance. Similarly, we can remove from T_0 paths containing a pattern described by $(a, \epsilon)(b, \epsilon)^*(a, a)$, which can be instead generated earlier via $(a, a)(b, \epsilon)^*(a, \epsilon)$. Figure 2 shows a transducer R describing

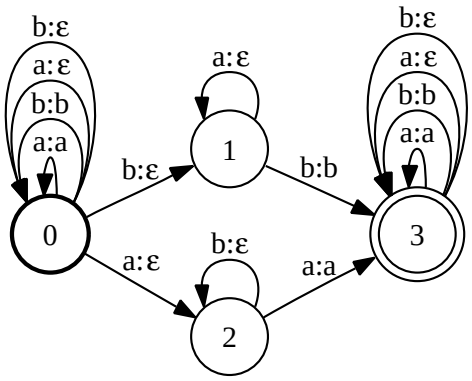


Fig. 2. Transducer R describing the set of paths to be removed from T_0.

the set of paths that we wish to remove from T_0.

To remove these paths, we can view R and T_0 as finite automata over the pair alphabet $(\Sigma \cup \{\epsilon\} \times \Sigma \cup \{\epsilon\}) - \{(\epsilon, \epsilon)\}$. We can thus use the standard automata complementation and difference algorithms to remove these paths [27]. The result is exactly the transducer T shown in Figure 1(b).

Theorem 8 *The transducer T maps each string x to the set of subsequences of x with exactly one occurrence of each.*

Proof. By construction, T maps each string x to a set of subsequences of x since it is derived from T_0 by removing some paths. No subsequence is lost since for each path of the form $(a, \epsilon)(b, \epsilon)^*(a, a)$ removed, there exists another path in T_0 generating the same output via $(a, a)(b, \epsilon)^*(a, \epsilon)$. Thus, T maps each string x to the set of all subsequences of x.

We now show that for any pair of input-output strings (x, y) accepted by T, there exists a unique path labeled with (x, y) in T. Fix a pair (x, y) accepted by T and let π_1 and π_2 be two paths labeled with (x, y). Let π be the longest prefix-path shared by π_1 and π_2.

π cannot end in state a or state b. Indeed, since these states are not final, there must be some suffix of (x, y) left to read. But, there is no input non-determinism at these states. The input symbol uniquely determines the transition to read. This contradicts the property of π being the longest common prefix-path.

Similarly, π cannot end in state F with some non-empty input symbol left to read since there is no input non-determinism at that state.

π cannot end in state I with some non-empty symbols left to read. Without loss of generality, assume that the input symbol is a. If the output symbol were also a, then the only alternative for the rest of both paths π_1 and π_2 at state I is the loop labeled with (a, a). But, that would contradict again the property of π being the longest common prefix-path. Similarly, if the output label is b, the only alternative for both paths is the transition from I to b followed by the one from b to I, again contradicting the status of π.

The only alternatives left are that π ends at state I or F with no other symbol left to read, that is $\pi = \pi_1 = \pi_2$. □

Corollary 3 *Let K be the subsequence kernel. Then, there exists a weighted transducer T over $(\mathbb{R}, +, \cdot, 0, 1)$ such that for all $x, y \in \Sigma^*$*

$$K(x, y) = (T \circ T^{-1})(x, y). \tag{36}$$

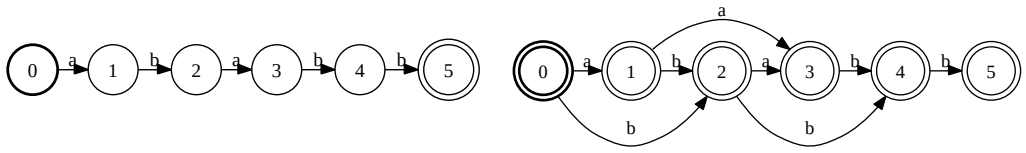
Proof. By Theorem 8, the (unweighted) transducer T maps each sequence to the set of its subsequences with multiplicity one. Let T' be the weighted transducer over $(\mathbb{R}, +, \cdot, 0, 1)$ derived from T by assigning weight 1 to all transitions and final weights. By definition of T, for all $x, y \in \Sigma^*$ such that y is a subsequence of x, $T'(x, y) = 1$, since there is a unique path in T labeled with (x, y). Thus, for all $x, y \in \Sigma^*$,

$$\begin{aligned}
(T' \circ T'^{-1})(x, y) &= \textstyle\sum_{u \in \Sigma^*} T'(x, u) T'(y, u) \\
&= \textstyle\sum_{u \sqsubseteq x, u \sqsubseteq y} T'(x, u) T'(y, u) \\
&= \textstyle\sum_{u \sqsubseteq x, u \sqsubseteq y} 1 \cdot 1 \\
&= K(x, y),
\end{aligned} \tag{37}$$

which ends the proof. □

The subsequence kernel K can thus be computed using the standard composition algorithm and shortest-distance algorithms [7]. The transducer T (or T') does not need to be computed beforehand. Instead, it can be determined on-demand, as needed for the specific strings x and y considered.

Since composition is associative, the composition operations for the computation of $X \circ T' \circ T'^{-1} \circ Y$, where X and Y are automata representing the strings x and y, can be carried out in any order [7]. In the specific case of the subsequence transducer T', it is advantageous to first compute $X \circ T'$ and $Y \circ T'^{-1}$. In fact, since after computation of $X \circ T'$, only the output labels of this transducer are needed, we can project it on the output, that is remove its input labels, and further optimize the result with the application of the



(a) (b)

Fig. 3. (a) Finite automaton X accepting the string $x = ababb$. (b) Finite automaton X' accepting exactly the set of subsequences of x obtained by application of ϵ-removal to the output projection of $X \circ T'$.

standard ϵ-removal algorithm [25]. It is not hard to see that the resulting finite automaton X' is a deterministic minimal automaton with the following properties:

(1) it has exactly as many states as X and all its states are final;
(2) it has at most $(|x| - 1)|\Sigma|$ transitions;
(3) it accepts exactly the set of subsequences of x with multiplicity one;
(4) it can be derived from X in the following simple manner: at any non-final state q of X and for any alphabet symbol c distinct from the one labeling the outgoing transition of q, create a new transition to q' with label c, where q' is the next following state along X with an incoming transition labeled with c. No transition is created when such a state does not exist.

All of these properties directly result from property 4). Figure 3 illustrates these properties in the case of a specific string. The application of ϵ-removal to the input projection of $Y \circ T'^{-1}$ results in an automaton Y' with the similar properties with respect to y.

$K(x, y)$ can be computed by applying a shortest-distance algorithm to compute the sum of the weights of all the paths of the automaton A resulting from the composition $X' \circ Y'$. The automaton A resulting from this composition admits at most $|X'||Y'| = |X||Y|$ states. Since both X' and Y' are deterministic, A is also deterministic with at most $|\Sigma|$ outgoing transitions at each state. Thus, the size of A or the cost of the composition $X' \circ Y'$ is in $O(|\Sigma||X||Y|)$. Since A is acyclic, a linear-time algorithm can be used to computed the sum of the weights of its paths [7].

It can be shown straightforwardly that the size of $X \circ T'$ is in $O(|\Sigma||X|)$. The cost of the application of ϵ-removal to compute X' from $X \circ T'$ is also in $O(|X \circ T'| + |X'|) = O(|\Sigma||X|)$ proceeding in reverse topological order to remove ϵ-transitions. Thus, the cost of the computation of X' is in $O(|\Sigma||X|)$ and similarly that of computing Y' in $O(|\Sigma||Y|)$. In view of that, the overall complexity of the computation of $K(x, y)$ is in $O(|\Sigma||x||y|)$. The computation of the subsequence kernel and other rational kernels can further benefit from a substantially more efficient algorithm for composing three or more transducers, *N-way composition* [1].

7 Conclusion

We introduced a new framework for learning languages that consists of mapping strings to a high-dimensional feature space and seeking linear separation in that space and applied this technique to the non-trivial case of PT languages and showed that this class of languages is indeed linearly separable and that the corresponding subsequence kernel can be computed efficiently. We further showed that the subsequence kernel is a positive definite symmetric rational kernel.

Many other classes of languages could be studied following the same ideas. This could lead to new results related to the problem of learning families of languages or classes of automata. Some preliminary analyses of linear separation with rational kernels suggests that kernels such as that the subsequence kernels with transducer values in a finite set admit a number of beneficial properties such as that of guaranteeing a positive margin [8].

Acknowledgements

Much of the work by Leonid Kontorovich was done while visiting the Hebrew University, in Jerusalem, Israel, in the summer of 2003. Many thanks to Yoram Singer for providing hosting and guidance at the Hebrew University. Thanks also to Daniel Neill and Martin Zinkevich for helpful discussions. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. The research at CMU was supported in part by NSF ITR grant IIS-0205456. This publication only reflects the authors' views.

The work of Mehryar Mohri was partially funded by a Google Research Award and the New York State Office of Science Technology and Academic Research (NYSTAR). This project was also sponsored in part by the Department of the Army Award Number W23RYX-3275-N605. The U.S. Army Medical Research Acquisition Activity, 820 Chandler Street, Fort Detrick MD 21702-5014 is the awarding and administering acquisition office. The content of this material does not necessarily reflect the position or the policy of the Government and no official endorsement should be inferred.

Appendix

A Linear Separability of Boolean Algebras

This section studies the linear separability of families of languages in greater abstraction.

Let $\mathcal{A} = \{A_i : i \in I\}$ denote a collection of languages $A_i \subseteq \Sigma^*$, which we shall refer to as *cover elements* and let $\mathrm{Bool}(\mathcal{A})$ denote the class of languages L that are finite Boolean combinations of the elements of $\mathcal{A}$. Let ψ be the natural embedding $\psi : \Sigma^* \to \{0,1\}^{\mathbb{N}}$ defined by

$$[\psi(x)]_i = [\![x \in A_i]\!]. \tag{A.1}$$

Define $\mathrm{LinSep}(\mathcal{A})$ to be the collection of languages $L \subset \Sigma^*$ that are finitely linearly separable under ψ. By Theorem 5, if $\mathcal{A}$ is a Regular Finite Cover then

$$\mathrm{LinSep}(\mathcal{A}) \subseteq \mathrm{Bool}(\mathcal{A}). \tag{A.2}$$

For the special case of $\mathcal{A} = \{Ш(u) : u \in \Sigma^*\}$, by Corollary 1, the following holds:

$$\mathrm{LinSep}(\mathcal{A}) \supseteq \mathrm{Bool}(\mathcal{A}). \tag{A.3}$$

For what other families $\mathcal{A}$ does the Property A.3 hold? A simple example shows that this property does not always hold. Let $\mathcal{A} = \{\emptyset, L_1, L_2, \Sigma^*\}$, $L_1 \neq L_2$ and L_1 and L_2 distinct from $\emptyset$ and Σ^*. Then, the language

$$L = L_1 \triangle L_2 = (L_1 \cup L_2) \setminus (L_1 \cap L_2) \tag{A.4}$$

is not linearly separable under ψ, in the same way as the function $\mathrm{XOR} : \{0,1\}^2 \to \{0,1\}$ is not linearly separable in $\mathbb{R}^2$.

The following theorem introduces three key properties that help generalize Theorem 2.

Theorem 9 *Let $\mathcal{A}$ be a family of languages verifying the following three properties:*

(1) Everywhere Dense Intersections (EDI): for any nonempty $A, B \in \mathcal{A}$, there is a nonempty $C \in \mathcal{A}$ such that

$$C \subseteq A \cap B. \tag{A.5}$$

(2) Finite Antichains (FAC): if $\mathcal{A}$ is partially ordered by set inclusion then any antichain must be finite.

(3) Locally Finite Cover (LFC): each $x \in \Sigma^*$ is contained in at most finitely many elements of $\mathcal{A}$.

Then, Property A.3 is satisfied: $\mathrm{LinSep}(\mathcal{A}) \supseteq \mathrm{Bool}(\mathcal{A})$.

Proof. (sketch) The proof is similar to that of Theorem 2. Using EDI, we can show as with the induction in Lemma 1 that any $L \in \mathrm{Bool}(\mathcal{A})$ admits a *decisive* $A \in \mathcal{A}$. Define such an A to be *maximally decisive* for L if $\mathcal{A}$ does not include an $A' \supsetneq A$ that is decisive for L (this corresponds to the definition of *minimally decisive* in the case of shuffle ideals).

We can use FAC to show that each $L \in \mathrm{Bool}(\mathcal{A})$ has finitely many maximally decisive cover elements. In the case of shuffle ideals, Higman's theorem was used to ensure that this property was satisfied.

If $V \in \mathrm{Bool}(\mathcal{A})$, then decisiveness modulo V is defined in the natural way and for any $L, V \in \mathrm{Bool}(\mathcal{A})$ there will be at least one but finitely many maximally decisive cover elements for L modulo V.

We follow the decision-list construction of Theorem 2, with $V_1 = \Sigma^*$ and

$$V_{i+1} = V_i \setminus \bigcup_{A \in \mathcal{D}_i} A, \tag{A.6}$$

where $\mathcal{D}$ is the set of the maximally decisive cover elements for L modulo V_i.

As in Theorem 2, we can show by contradiction that this process terminates. Suppose the algorithm generated an infinite sequence of maximally decisive sets: $\mathcal{D}_1, \mathcal{D}_2, \ldots$ Construct an infinite sequence $(X_n)_{n \in \mathbb{N}}$ by selecting a cover element $X_n \in \mathcal{D}_n$, for any $n \in \mathbb{N}$. By construction, we cannot have

$$X_m \subseteq X_n, \qquad m > n. \tag{A.7}$$

Thus, in particular, all the sets X_n are distinct. As previously, we define the new sequence $(Y_n)_{n \in \mathbb{N}}$ by $Y_1 = X_1$ and $Y_{n+1} = X_{\xi(n)}$, where $\xi : \mathbb{N} \to \mathbb{N}$ is given by

$$\xi(n) = \begin{cases} \min\{k \in \mathbb{N} : \{Y_1, \ldots, Y_n, X_k\} \text{ is an antichain}\}, & \text{if such a } k \text{ exists,} \\ \infty & \text{otherwise.} \end{cases} \tag{A.8}$$

We cannot have $\xi(n) \neq \infty$ for all $n > 0$ since the set $\{Y_1, Y_2, \ldots\}$ would then be an infinite antichain, violating FAC. Thus, $\xi(n) = \infty$ for some $n > 0$, and our sequence of Y's is finite: $\mathcal{Y} = \{Y_1, Y_2, \ldots, Y_N\}$. Since A.7 does not hold, it follows that for $k > N$, each X_k contains some $Y \in \mathcal{Y}$, which violates LFC. This shows that the decision list generated is indeed finite. Verifying its correctness is very similar to the inductive argument used in the proof of Theorem 2. □

A particularly intriguing problem, which we leave open for now, is that of providing an exact characterization of the families of languages $\mathcal{A}$ for which the equality $\mathrm{LinSep}(\mathcal{A}) = \mathrm{Bool}(\mathcal{A})$ holds.

B Linear Separability of Regular Languages

Our study of linear separability of languages naturally raises the question of whether the family of *all* regular languages is finitely linearly separable under some universal embedding. It turns out that there exists indeed a *universal regular* kernel $K_{\text{UNIV}} : \Sigma^* \times \Sigma^* \to \mathbb{R}$ for which all regular languages are linearly separable [19].

Consider the set of deterministic finite automata (DFAs) over a fixed alphabet Σ. Let $L(M)$ denote the regular language accepted by a DFA M and let $\text{DFA}(n)$ denote the set of all DFAs with n states. Our universal kernel is based on the auxiliary kernel K_n:

$$K_n(x, y) = \sum_{M \in \text{DFA}(n)} [\![x \in L(M)]\!] \, [\![y \in L(M)]\!]. \tag{B.1}$$

Thus, K_n counts the number of DFAs with n states that accept both x and y. The universal kernel is then defined by [19]:

$$K_{\text{UNIV}}(x, y) = [\![x = y]\!] + \sum_{n=1}^{\min\{|x|,|y|\}} K_n(x, y). \tag{B.2}$$

The following theorem shows the universal separability property of that kernel [19].

Theorem 10 *Every regular language is finitely linearly separable under the embedding corresponding to* K_{UNIV}.

This embedding, modulo the fact that it is defined in terms of a direct sum of two embeddings [19], corresponds to the family of sets $\mathcal{A}$, where each $A \in \mathcal{A}$ is of the form

$$A = \{x \in L(M) : M \in \text{DFA}(n), 1 \leq n \leq |x|\}. \tag{B.3}$$

It is not hard to verify that $\mathcal{A}$ is a Regular Finite Cover. Thus, the converse of Theorem 10 is also valid: any language separated by K_{UNIV} is regular. Combining these observations with the Representer Theorem 6 yields the following characterization of regular languages.

Theorem 11 *A language* $L \subseteq \Sigma^*$ *is regular if and only if there is a finite number of* support strings $s_1, \ldots, s_m \in \Sigma^*$ *and weights* $\alpha_1, \ldots, \alpha_m \in \mathbb{R}$ *such that*

$$L = \{x \in \Sigma^* : \sum_{i=1}^{m} \alpha_i K_{\text{UNIV}}(s_i, x) > 0\}. \tag{B.4}$$

Since K_{UNIV} linearly separates all regular languages, *a fortiori*, it also linearly separates the PT languages. However, while the subsequence kernel used to separate PT languages was shown to admit an efficient computation (Section 6), K_{UNIV} is not known to enjoy the same property (an efficient approximation method is presented in [19] however). Also, the margins obtained by using K_{UNIV} are likely to be significantly worse than those resulting from the subsequence kernel. However, we have not yet derived quantitative margin bounds for the universal kernel that could enable this comparison.

References

[1] Cyril Allauzen and Mehryar Mohri. N-Way Composition of Weighted Finite-State Transducers. Technical Report TR2007-902, Courant Institute of Mathematical Sciences, New York University, August 2007.

[2] Dana Angluin. On the complexity of minimum inference of regular sets. *Information and Control*, 3(39):337–350, 1978.

[3] Dana Angluin. Inference of reversible languages. *Journal of the ACM (JACM)*, 3(29):741–765, 1982.

[4] Martin Anthony. Threshold Functions, Decision Lists, and the Representation of Boolean Functions. Neurocolt Technical report Series NC-TR-96-028, Royal Holloway, University of London, 1996.

[5] Peter Bartlett and John Shawe-Taylor. Generalization performance of support vector machines and other pattern classifiers. In *Advances in kernel methods: support vector learning*, pages 43–54. MIT Press, Cambridge, MA, USA, 1999.

[6] Bernhard E. Boser, Isabelle Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop of Computational Learning Theory*, volume 5, pages 144–152, Pittsburg, 1992. ACM.

[7] Corinna Cortes, Patrick Haffner, and Mehryar Mohri. Rational Kernels: Theory and Algorithms. *Journal of Machine Learning Research (JMLR)*, 5:1035–1062, 2004.

[8] Corinna Cortes, Leonid Kontorovich, and Mehryar Mohri. Learning Languages with Rational Kernels. In *Proceedings of The 20th Annual Conference on Learning Theory (COLT 2007)*, volume 4539 of *Lecture Notes in Computer Science*, pages 349–364, San Diego, California, June 2007. Springer, Heidelberg, Germany.

[9] Corinna Cortes and Vladimir N. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995.

[10] Jonathan Derryberry, 2004. private communication.

[11] Yoav Freund, Michael Kearns, Dana Ron, Ronitt Rubinfeld, Robert E. Schapire, and Linda Sellie. Efficient learning of typical finite automata from random walks. In *STOC '93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 315–324, New York, NY, USA, 1993. ACM Press.

[12] Pedro García and José Ruiz. Learning k-testable and k-piecewise testable languages from positive data. *Grammars*, 7:125–140, 2004.

[13] E. Mark Gold. Language identification in the limit. *Information and Control*, 50(10):447–474, 1967.

[14] E. Mark Gold. Complexity of automaton identification from given data. *Information and Control*, 3(37):302–420, 1978.

[15] L. H. Haines. On free monoids partially ordered by embedding. *Journal of Combinatorial Theory*, 6:35–40, 1969.

[16] David Haussler, Nick Littlestone, and Manfred K. Warmuth. Predicting $\{0, 1\}$-Functions on Randomly Drawn Points. In *Proceedings of the first annual workshop on Computational learning theory (COLT 1988)*, pages 280–296, San Francisco, CA, USA, 1988. Morgan Kaufmann Publishers Inc.

[17] George Higman. Odering by divisibility in abstract algebras. *Proceedings of The London Mathematical Society*, 2:326–336, 1952.

[18] Micheal Kearns and Umesh Vazirani. *An Introduction to Computational Learning Theory*. The MIT Press, 1997.

[19] Leonid Kontorovich. A Universal Kernel for Learning Regular Languages. In *The 5th International Workshop on Mining and Learning with Graphs (MLG 2007)*, Florence, Italy, 2007.

[20] Leonid Kontorovich, Corinna Cortes, and Mehryar Mohri. Learning Linearly Separable Languages. In *Proceedings of The 17th International Conference on Algorithmic Learning Theory (ALT 2006)*, volume 4264 of *Lecture Notes in Computer Science*, pages 288–303, Barcelona, Spain, October 2006. Springer, Heidelberg, Germany.

[21] Werner Kuich and Arto Salomaa. *Semirings, Automata, Languages*. Number 5 in EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1986.

[22] Huma Lodhi, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. Text classification using string kernels. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *NIPS 2000*, pages 563–569. MIT Press, 2001.

[23] M. Lothaire. *Combinatorics on Words*, volume 17 of *Encyclopedia of Mathematics and Its Applications*. Addison-Wesley, 1983.

[24] Alexandru Mateescu and Arto Salomaa. *Handbook of Formal Languages, Volume 1: Word, Language, Grammar*, chapter Formal languages: an Introduction and a Synopsis, pages 1–39. Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[25] Mehryar Mohri. Generic Epsilon-Removal and Input Epsilon-Normalization Algorithms for Weighted Transducers. *International Journal of Foundations of Computer Science*, 13(1):129–143, 2002.

[26] José Oncina, Pedro García, and Enrique Vidal. Learning subsequential transducers for pattern recognition interpretation tasks. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(5):448–458, 1993.

[27] Dominique Perrin. Finite automata. In J. Van Leuwen, editor, *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, pages 1–57. Elsevier, Amsterdam, 1990.

[28] Leonard Pitt and Manfred Warmuth. The minimum consistent DFA problem cannot be approximated within any polynomial. *Journal of the Assocation for Computing Machinery*, 40(1):95–142, 1993.

[29] Dana Ron, Yoram Singer, and Naftali Tishby. On the learnability and usage of acyclic probabilistic finite automata. *Journal of Computer and System Sciences*, 56(2):133–152, 1998.

[30] Imre Simon. Piecewise testable events. In *Automata Theory and Formal Languages*, pages 214–222, 1975.

[31] Boris A. Trakhtenbrot and Janis M. Barzdin. *Finite Automata: Behavior and Synthesis*, volume 1 of *Fundamental Studies in Computer Science*. North-Holland, Amsterdam, 1973.

[32] Vladimir N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.