Learning Languages with Rational Kernels

Corinna Cortes[1], Leonid Kontorovich[2], and Mehryar Mohri[3,1]

[1] Google Research,
76 Ninth Avenue, New York, NY 10011.
[2] Carnegie Mellon University,
5000 Forbes Avenue, Pittsburgh, PA 15213.
[3] Courant Institute of Mathematical Sciences,
251 Mercer Street, New York, NY 10012.

Abstract. We present a general study of learning and linear separability with rational kernels, the sequence kernels commonly used in computational biology and natural language processing. We give a characterization of the class of all languages linearly separable with rational kernels and prove several properties of the class of languages linearly separable with a fixed rational kernel. In particular, we show that for kernels with finite range, these languages are necessarily finite Boolean combinations of preimages by a transducer of a single sequence. We also analyze the margin properties of linear separation with rational kernels and show that kernels with finite range guarantee a positive margin and lead to better learning guarantees. Creating a finite range rational kernel is often non-trivial even for relatively simple cases. However, we present a novel and general algorithm, double-tape disambiguation, that takes as input an arbitrary transducer mapping sequences to sequence features, and yields an associated transducer that defines a finite range rational kernel. We describe the algorithm in detail and show its application to several cases of interest.

1 Motivation

In previous work, we introduced a paradigm for learning languages that consists of mapping strings to an appropriate high-dimensional feature space and learning a separating hyperplane in that space [13]. We proved that the rich class of piecewise-testable languages [21] can be linearly separated using a high-dimensional feature mapping based on subsequences. We also showed that the positive definite kernel associated to this embedding, the *subsequence kernel*, can be efficiently computed. Support vector machines can be used in combination with this kernel to determine a separating hyperplane for piecewise-testable languages. We further proved that the languages linearly separable with this kernel are exactly the piecewise-testable languages.

The subsequence kernel is a *rational kernel* – that is, a kernel that can be represented by weighted finite-state transducers [5, 12]. Most sequence kernels successfully used in computational biology and natural language processing, including mismatch kernels [15], gappy n-gram kernels [16], locality-improved kernels [24], convolutions kernels for strings [11], tree kernels [4], n-gram kernels [5], and moment kernels [6], are special instances of rational kernels. Rational kernels can be computed in quadratic time using a single general algorithm [5].

This motivates our study of learning with rational kernels, in particular the question of determining the class of languages that can be linearly separated with a given rational kernel, thereby generalizing the result relating to subsequence kernels and piecewise-testable languages, and also analyzing their generalization properties based on the margin. It is also natural to ask which languages are separable with rational kernels in general.

This paper deals with precisely these questions. We prove that the family of languages linearly separable with rational kernels is exactly that of *stochastic languages* [20], a class of languages that strictly includes regular languages and contains non-trivial context-free and context-sensitive languages. We also prove several properties of the class of languages linearly separable with a fixed rational kernel. In particular, we show that when the kernel has finite range these languages are necessarily finite Boolean combinations of the preimages by a transducer of single sequences.

In previous work, we proved that linear separability with the subsequence kernel guarantees a positive margin, which helped us derive margin-based bounds for learning piecewise-testable languages [13]. This property does not hold for all rational kernels. We prove however that a positive margin is guaranteed for all rational kernels with finite range.

This quality and the property of the languages they separate in terms of finite Boolean combinations point out the advantages of using PDS rational kernels with finite range, such as the subsequence kernel used for piecewise-testable languages. However, while defining a transducer mapping sequences to the *feature sequences* of interest is typically not hard, creating one that associates to each sequence at most a predetermined finite number of instances of that feature is often non-trivial, even for relatively simple transducers.

We present a novel algorithm, *double-tape disambiguation*, to precisely address this problem. The algorithm takes as input an (unweighted) arbitrary transducer mapping sequences to features and yields a transducer associating the same features to the same input sequences but at most once. The algorithm can thus help define and represent rational kernels with finite range, which offer better learning guarantees. We describe the algorithm in detail and show its application to several cases of interest.

The paper is organized as follows. Section 2 introduces the definitions and notation related to weighted transducers and probabilistic automata that are used in the remainder of the paper. Section 3 gives the proof of several characterization theorems for the classes of languages that can be linearly separated with rational kernels. The margin properties of rational kernels are studied in Section 4. Section 5 describes in detail the double-tape disambiguation algorithm which can be used to define complex finite range rational kernels and shows its application to several cases of interest.

2 Preliminaries

This section gives the standard definition and specifies the notation used for weighted transducers and briefly summarizes the definition and essential properties of probabilistic automata, which turn out to play an important role in our study of linear separability with rational kernels.

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 $\overline{L} = \Sigma^* \setminus L$. We also denote by $|x|_a$ the number of occurrences of the symbol a in x.

2.1 Weighted Transducers

Finite-state transducers are finite automata in which each transition is augmented with an output label in addition to the familiar input label [2, 10]. Output labeled are concatenated along a path to form an output sequence as with input labels. *Weighted transducers* are finite-state transducers in which each transition carries some weight in addition to the input and output labels. The

weights of the weighted transducers considered in this paper are real values and are multiplied along the paths. The weight of a pair of input and output strings (x, y) is obtained by summing the weights of the paths labeled with (x, y). The following gives a formal definition of weighted transducers. In all the following definitions $\mathbb{K}$ denotes either the set of real numbers $\mathbb{R}$, rational numbers $\mathbb{Q}$, or integers $\mathbb{Z}$.

Definition 1. *A* weighted finite-state transducer T *over* $(\mathbb{K}, +, \cdot, 0, 1)$ *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,* $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times \mathbb{K} \times Q$ *a finite set of transitions,* $\lambda : I \to \mathbb{K}$ *the initial weight function, and* $\rho : F \to \mathbb{K}$ *the final weight function mapping* F *to* $\mathbb{K}$.

For a path π in a transducer, we denote by $p[\pi]$ the origin state of that path and by $n[\pi]$ its destination state. We also denote by $P(I, x, y, F)$ the set of paths from the initial states I to the final states F labeled with input string x and output string y. A transducer T is *regulated* if the output weight associated by T to any pair of input-output strings (x, y) by:

$$T(x, y) = \sum_{\pi \in P(I,x,y,F)} \lambda(p[\pi]) \cdot w[\pi] \cdot \rho[n[\pi]] \tag{1}$$

is well-defined and in $\mathbb{K}$. $T(x, y) = 0$ when $P(I, x, y, F) = \emptyset$. If for all $q \in Q$ $\sum_{\pi \in P(q,\epsilon,\epsilon,q)} w[\pi] \in \mathbb{K}$, then T is regulated. In particular, when T does not admit any ϵ-cycle, it is regulated. The weighted transducers we will be considering in this paper will be regulated. Figure 1(a) shows an example of weighted transducer.

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 matching input and output alphabets Σ, is a weighted transducer denoted by $T_1 \circ T_2$ when the sum:

$$(T_1 \circ T_2)(x, y) = \sum_{z \in \Sigma^*} T_1(x, z) \cdot T_2(z, y) \tag{2}$$

is well-defined and in $\mathbb{K}$ for all $x, y \in \Sigma^*$ [14].

Weighted automata can be defined as weighted transducers T with identical input and output labels, that is $T(x, y) = 0$ for $x \neq y$. Equivalently, since output and input labels of all paths coincide, output (or input) labels can be omitted. Thus, weighted automata can also be derived from weighted transducers by omitting output labels. Thus the weight $A(x)$ assigned by a weighted automaton A to a sequence $x \in \Sigma^*$ is, when it is well-defined and in $\mathbb{K}$,

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

where $P(I, x, F)$ is the set of paths from an initial state to a final state labeled with x. Figure 1(b) shows and example of weighted automaton.

2.2 Probabilistic Automata

In this paper, we will consider *probabilistic automata* as originally defined by by M. Rabin [19, 18].

Fig. 1. (a) Example of weighted transducer T. (b) Example of weighted automaton A. A can be obtained from T by projection on the input. A bold circle indicates an initial state and a double-circle a final state. A final state carries a weight indicated after the slash symbol representing the state number. The initial weights are not indicated in all the examples in this paper since they are all equal to one.

Definition 2. *A weighted automaton A over $\mathbb{K}$ is said to be* probabilistic *if its weights are non-negative, if it admits no ϵ-transition, and if at each state, the weights of the outgoing transitions labeled with the same symbol sum to one.*

Thus, a probabilistic automaton in this sense defines a conditional probability distribution $\Pr[q' \mid q, x]$ over all states q' that can be reached from q by reading a sequence x.[4] Probabilistic automata can be used to define languages as follows.

Definition 3 ([19]). *A language L is said to be* $\mathbb{K}$-stochastic *if there exist a probabilistic automaton A and $\lambda \in \mathbb{K}$, $\lambda > 0$, such that $L = \{x : A(x) > \lambda\}$. λ is then called a* cut-point.

Note that stochastic languages are not necessarily regular. They include non-trivial classes of context-free and context-sensitive languages.[5] A cut-point λ is said to be *isolated* if there exists $\delta > 0$ such that $\forall x \in \Sigma^*,\ 0 < \delta \leq |A(x) - \lambda|$. Rabin [19] showed that when λ is an *isolated cut-point*, then the stochastic language defined as above is regular.

3 Properties of Linearly Separated Languages

This section analyzes the properties of the languages separated by rational kernels. It presents a characterization of the set of all languages linearly separable with rational kernels and analyzes the properties of these languages for a fixed rational kernel.

3.1 Rational Kernels

A general definition of rational kernels based on weighted transducers defined over arbitrary semirings was given by [5]. The following is a simpler definition for the case of transducers defined over $(\mathbb{K}, +, \cdot, 0, 1)$ that we consider here.

A string kernel $K : \Sigma^* \times \Sigma^* \to \mathbb{K}$ is *rational* if it coincides with the function defined by a weighted transducer U over $(\mathbb{K}, +, \cdot, 0, 1)$, that is for all $x, y \in \Sigma^*,\ K(x, y) = U(x, y)$.

[4] This definition of probabilistic automata differs from another one commonly used in language modeling and other applications (see for example [7]) where A defines a probability distribution over all strings. With that definition, A is probabilistic if for any state $q \in Q$, $\sum_{\pi \in P(q,q)} w[\pi]$, the sum of the weights of all cycles at q, is well-defined and in $\mathbb{R}_+$ and $\sum_{x \in \Sigma^*} A(x) = 1$.

[5] We are using here the original terminology of *stochastic languages* used in formal language theory [20]. Some authors have recently used the same terminology to refer to completely different families of languages [9].

Not all rational kernels are *positive definite and symmetric* (PDS), or equivalently verify the Mercer condition, a condition that guarantees the convergence of training for discriminant classification algorithms such as SVMs. But, for any weighted transducer T over $(\mathbb{K}, +, \cdot, 0, 1)$, $U = T \circ T^{-1}$ is guaranteed to define a PDS kernel [5]. Conversely, it was conjectured that PDS rational kernels coincide with the transducers U of the form $U = T \circ T^{-1}$. A number of proofs related to closure properties favor this conjecture [5]. Furthermore, most rational kernels used in computational biology and natural language processing are of this form [15, 16, 24, 4, 6, 5]. To ensure the PDS property, we will consider in what follows only rational kernels of this form.

Our paradigm for learning languages is based on a linear separation using PDS kernels. We will say that a language $L \subseteq \Sigma^*$ is *linearly separable by a kernel* K, if there exist $b \in \mathbb{K}$ and a finite number of strings $x_1, \ldots, x_m \in \Sigma^*$ and elements of $\mathbb{K}$, $\alpha_1, \ldots, \alpha_m \in \mathbb{K}$, such that

$$L = \{x : \sum_{i=1}^{m} \alpha_i K(x_i, x) + b > 0\}. \tag{4}$$

Lemma 1. *A language $L \subseteq \Sigma^*$ is linearly separable by a rational kernel $K = T \circ T^{-1}$ iff there exists an acyclic weighted automaton A and $b \in \mathbb{K}$ such that*

$$L = \{x : A \circ (T \circ T^{-1}) \circ M_x + b > 0\}, \tag{5}$$

where M_x is a finite (unweighted) automaton representing the string x.

Proof. When K is a rational kernel, $K = T \circ T^{-1}$, the linear combination defining the separating hyperplane can be written as:

$$\sum_{i=1}^{m} \alpha_i K(x_i, x) = \sum_{i=1}^{m} \alpha_i (T \circ T^{-1})(x_i, x) = \sum_{i=1}^{m} \alpha_i (M_{x_i} \circ T \circ T^{-1} \circ M_x) \tag{6}$$

$$= (\sum_{i=1}^{m} \alpha_i M_{x_i}) \circ T \circ T^{-1} \circ M_x, \tag{7}$$

where we used the distributivity of $+$ over composition, that is for any three weighted transducers $(U_1 \circ U_3) + (U_2 \circ U_3) = (U_1 + U_2) \circ U_3$ (a consequence of distributivity and of $+$ over $\times$ and the definition of composition). The result follows the observation that a weighted automaton A over $(\mathbb{K}, +, \cdot, 0, 1)$ is acyclic iff it is equivalent to $\sum_{i=1}^{m} \alpha_i M_{x_i}$ for some strings $x_1, \ldots, x_m \in \Sigma^*$ and elements of $\mathbb{K}$, $\alpha_1, \ldots, \alpha_m \in \mathbb{K}$. □

3.2 Languages Linearly Separable with Rational Kernels

This section presents a characterization of the languages linearly separable with rational kernels.

Theorem 1. *A language L is linearly separable by a PDS rational kernel $K = T \circ T^{-1}$ iff it is stochastic.*

Proof. Assume that L is linearly separable by a rational kernel and let T be a weighted transducer such that $K = T \circ T^{-1}$. By lemma 1, there exist $b \in \mathbb{K}$ and an acyclic weighted automaton A such that $L = \{x : A \circ (T \circ T^{-1}) \circ M_x + b > 0\}$. Let R denote the projection of the weighted transducer $A \circ T \circ T^{-1}$ on the output, that is the weighted automaton over $(\mathbb{K}, +, \cdot, 0, 1)$ derived from $A \circ T \circ T^{-1}$ by omitting input labels. Then, $A \circ (T \circ T^{-1}) \circ M_x = R \circ M_x = R(x)$. Let S be

the weighted automaton $R + b$, then, $L = \{x : S(x) > 0\}$. By Turakainen's theorem ([22, 20]), a language defined in this way is stochastic, which proves one direction of the theorem's claim.

Conversely, let R be a probabilistic automaton and $\lambda \in \mathbb{K}$, $\lambda > 0$, such $L = \{x : R(x) > \lambda\}$. We can assume $L \neq \emptyset$ since any rational kernel can be trivially used to linearly separate the empty set by using an empty acyclic automaton A. It is straightforward to construct a weighted automaton R_λ assigning weight λ to all strings in Σ^*. Let S denote the weighted automaton over $(\mathbb{K}, +, \cdot, 0, 1)$ defined by $S = R - R_\lambda$. Thus, $L = \{x : S(x) > 0\}$. Let T be the weighted transducer constructed from S by augmenting all transitions of S with the same output label ϵ. By construction, for all $x, y \in \Sigma^*$, $T(x, y) = S(x)$ if $y = \epsilon$, $T(x, y) = 0$ otherwise and

$$(T \circ T^{-1})(x, y) = \sum_{z \in \Sigma^*} T(x, z)T(y, z) = T(x, \epsilon)T(y, \epsilon) = S(x) \cdot S(y). \tag{8}$$

Since $L \neq \emptyset$, we can select an arbitrary string $x_0 \in L$, thus $S(x_0) > 0$. Let A be the acyclic automaton only accepting the string x_0 and with weight 1. Then,

$$\forall x \in \Sigma^*,\ A \circ (T \circ T^{-1}) \circ M_x = A(x_0) \cdot (T \circ T^{-1})(x_0, x) = S(x_0) \cdot S(x). \tag{9}$$

Since $S(x_0) > 0$, $A \circ (T \circ T^{-1}) \circ M_x > 0$ iff $S(x) > 0$, which proves that L can be linearly separated with a PDS rational kernel. □

The theorem highlights the importance of stochastic languages to the question of linear separation of languages with rational kernels. The proof is constructive. Given a PDS rational kernel $K = T \circ T^{-1}$, $b \in \mathbb{K}$, and an acyclic automaton A, a probabilistic automaton B can be constructed and a cut-off $\lambda \in \mathbb{K}$ determined such that:

$$L = \{x : A \circ (T \circ T^{-1}) \circ M_x + b > 0\} = \{x : B(x) > \lambda\}, \tag{10}$$

using the weighted automaton S derived from T, b, and A as in the proof of Theorem 1, and the following result due to Turakainen [22].

Theorem 2 ([22]). *Let S be a weighted automaton over $(\mathbb{K}, +, \cdot, 0, 1)$ with n states, with $\mathbb{K} = \mathbb{R}$ or $\mathbb{K} = \mathbb{Q}$. A probabilistic automaton B over $(\mathbb{K}, +, \cdot, 0, 1)$ with $n + 3$ states can be constructed from S such that:*

$$\forall x \in \Sigma^+,\ S(x) = c^{|x|} \left(B(x) - \frac{1}{n+3} \right), \tag{11}$$

where $c \in \mathbb{K}$ is a large number.

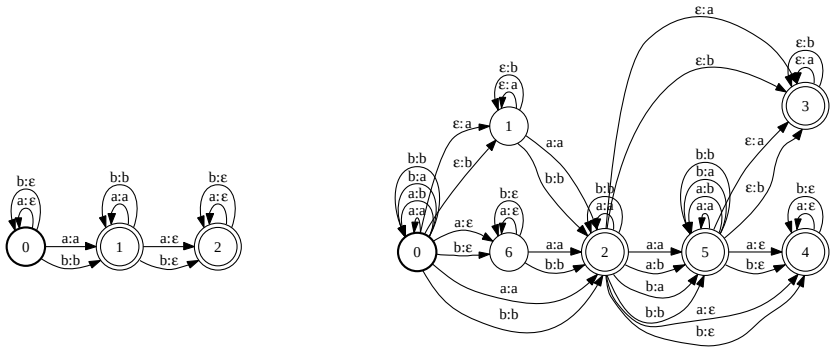
3.3 Family of Languages Linearly Separable with a Fixed Rational Kernel

Theorem 1 provides a characterization of the set of linearly separable languages with rational kernels. This section studies the family of languages linearly separable by a given PDS rational kernel $K = T \circ T^{-1}$.

A weighted transducer T defines an (unweighted) mapping from Σ^* to Σ^* (a *transduction*) denoted by $\hat{T}$:

$$\forall x \in \Sigma^*,\ \hat{T}(x) = \{y : T(x, y) \neq 0\}. \tag{12}$$

For any $x \in \Sigma^*$, $\hat{T}(x)$ is a regular language that can be computed from a weighted transducer T by projecting $M_x \circ T$ on the output side, applying weighted determinization, and then removing the weights.



(a) (b)

Fig. 2. (a) Weighted transducer T counting the number of occurrences of non-empty substrings of any length: for each $x \in \Sigma^*$ and any substring $y \in \Sigma^+$ of x, $T(x, y)$ gives the number of occurrences of y in x. All transition weights and final weights are equal to 1. (b) Corresponding kernel transducer $K = T \circ T^{-1}$.

$\hat{T}(x)$ can be viewed as the set of non-zero features y (sequences) associated to x by the kernel K, each with some weight $T(x, y)$. For example, for the kernel of Figure 2(b), $\hat{T}(x)$ associates to x the set of its substrings, that is contiguous sequences of symbols appearing in x.

For all the rational kernels we have seen in practice, the cardinality of $\hat{T}(x)$ is finite for any $x \in \Sigma^*$. $\hat{T}(x)$ may be for example the set of substrings, n-grams, or other subsequences, which in all cases are finite. Furthermore, when $\hat{T}(x)$ is not finite, then T is typically not a regulated transducer. This justifies the assumption made in the following theorem.

Theorem 3. *Let $K = T \circ T^{-1}$ be a PDS rational kernel. Assume that for each $x \in \Sigma^*$, $\hat{T}(x)$ is finite. Then, a language L linearly separable by K is necessarily of the form*

$$L = \{x : \sum_{i=1}^{n} \lambda_i T(x, z_i) + b > 0\}, \tag{13}$$

with $z_1, \ldots, z_n \in \Sigma^*$ and $\lambda_1, \ldots, \lambda_n, b \in \mathbb{K}$.

Proof. Let L be a language linearly separable by K. By Lemma 1, there exists an acyclic weighted automaton A and $b \in \mathbb{K}$ such that $L = \{x : A \circ (T \circ T^{-1}) \circ M_x + b > 0\}$, where M_x is a finite automaton representing the string x. Since $\hat{T}(x)$ is finite and A is acyclic, $\bigcup_{x:A(x)\neq 0}\{y : (A \circ T)(x, y) \neq 0\}$ is a finite set. Thus, the projection of $(A \circ T)$ on the output side is an acyclic weighted automaton and is thus equivalent to $\sum_{i=1}^{n} \lambda_i M_{z_i}$ for some $z_1, \ldots, z_n \in \Sigma^*$ and $\lambda_1, \ldots, \lambda_n, b \in \mathbb{K}$. By definition of L,

$$L = \{x : \sum_{i=1}^{n} \lambda_i \; M_{z_i} \circ T^{-1} \circ M_x + b\} = \{x : \sum_{i=1}^{n} \lambda_i T(x, z_i) + b > 0\}. \tag{14}$$

□

Corollary 1. *Let $K = T \circ T^{-1}$ be a PDS rational kernel. Assume that for each $x \in \Sigma^*$, $\hat{T}(x)$ is finite and that $T(x, y)$ takes values in some finite $E_T \subset \mathbb{K}$. Then, the following two properties hold:*

1. *L is necessarily of the form:*

$$L = \bigcup_{0 \neq v \in E_T} \{x : \sum_{i=1}^{n} \lambda_i^v \mathbb{1}_{\left\{x \in \hat{T}_v^{-1}(z_i^v)\right\}} + b^v > 0\}, \tag{15}$$

where $\hat{T}_v(x) = \{y : T(x,y) = v\}$ *and* $z_1^v, \ldots, z_n^v \in \Sigma^*$ *and* $\lambda_1^v, \ldots, \lambda_n^v, b^v \in \mathbb{K}$*;*

2. *L is a finite Boolean combination of languages $L_z = \hat{T}^{-1}(z)$.*

Proof. The first assertion follows directly from Theorem 3 and the definition of $\hat{T}$. For $v \in E_T$, let the function $f_v : \Sigma^* \to \mathbb{K}$ be defined by

$$f_v(x) = \sum_{i=1}^{n} \lambda_i^v \mathbb{1}_{\left\{x \in \hat{T}_v^{-1}(z_i)\right\}} + b^v, \tag{16}$$

and observe that f_v must have a finite range $\{r_k^v \in \mathbb{K} : k = 1, \ldots, K^v\}$, where $K^v \leq 2^n$. Let $L_{r_k^v} \subseteq \Sigma^*$ be defined by

$$L_{r_k^v} = f_v^{-1}(r_k). \tag{17}$$

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

$$L_I^v = \left(\bigcap_{i \in I} \hat{T}_v^{-1}(z_i^v) \right) \setminus \left(\bigcup_{i \in \{1,\ldots,N\} \setminus I} \hat{T}_v^{-1}(z_i^v) \right). \tag{18}$$

Each $L_{r_k^v}$ is the union of finitely many r_k^v-acceptable L_I^v's, and L is the union of the $L_{r_k^v}$ for positive r_k^v. □

The Corollary provides some insight about the family of languages that are linearly separable with a fixed PDS rational kernel $K = T \circ T^{-1}$ with finite range. In practice, it is often straightforward to determine $\hat{T}_v^{-1}(x)$ for any $x \in \Sigma^*$. For example, for the subsequence kernel, $\hat{T}^{-1}(x)$ represents the set of all sequences admitting x as a subsequence. Corollary 1 shows that any language linearly separated by K is a finite Boolean combination of these sets. This result and that of Theorem 3 apply to virtually all cases in computational biology or natural language processing where string kernels are used in combination with SVMs, since most string kernels used in practice (if not all) are rational kernels.

It was proven by [13] that in the case of the subsequence kernel, the second property of the Corollary 1 represents in fact a characterization of linearly separable languages (piecewise-testable languages). In general, however, the converse may not hold. There exist indeed finite boolean combinations of $\hat{T}^{-1}(x)$ that are not linearly separable when K is selected to be the n-gram kernel for example.

Corollary 1 points out an interesting property of PDS rational kernels with finite range. In the following section, we will see that linear separability with such kernels also ensures useful margin properties.

4 Learning and Margin Guarantees

This section deals with the problem of learning families of languages using PDS rational kernels. Linear separability with some rational kernels K guarantees a positive margin. In particular, as previously shown, the subsequence kernel guarantees a positive margin [13].

When this property holds, a linear separation learning technique such as support vector machines (SVMs) [3, 8, 23] combined with a rational kernel K can be used to learn a family of

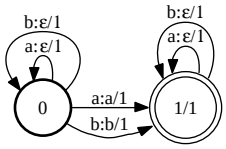


Fig. 3. Weighted transducer T counting the number of occurrences of as and bs: $\forall x \in \Sigma^*, T(x,a) = |x|_a, T(x,b) = |x|_b$.

languages. Since rational kernels can be computed in quadratic time [5], the complexity of the algorithm for a sample of size m where $x_{\max}$ is the longest string is in $O(\mathrm{QP}(m))+m^2\,|x_{\max}|^2\,|\Sigma|)$, 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 when that margin property holds. Note however that since the VC-dimension of the typical family of languages one wishes to learn is infinite, e.g., piecewise testable languages, PAC-learning is not possible and we need to resort to a weaker guarantee.

Not all PDS rational kernels guarantee a positive margin (as we shall see later), but we will prove that all PDS rational kernels with finite range admit this property, which further emphasizes their benefits for learning.

4.1 Margin

Let $\mathcal{S}$ be a sample extracted from a set X ($X = \Sigma^*$ when learning languages) and let the margin ρ of a hyperplane with weight vector $w \in \mathbb{K}^{\mathbb{N}}$ and offset $b \in \mathbb{K}$ over this sample be defined by:

$$\rho = \inf_{x \in \mathcal{S}} \frac{|\langle w, \Phi(x)\rangle + b|}{\|w\|}.$$

This definition also holds for infinite-size samples. For finite samples, linear separation with a hyperplane $\langle w, x\rangle + b = 0$ is equivalent to a positive margin $\rho > 0$. But, this may not hold for infinite-size samples, since points in an infinite-dimensional space may be arbitrarily close to the separating hyperplane and their infimum distance could be zero. There are in fact PDS rational kernels for which this can occur.

4.2 Example of Linear Separation with Zero Margin

Let $K = T \circ T^{-1}$ be the PDS rational defined by the weighted transducer T counting the number of occurrences of a and b when the alphabet $\Sigma = \{a, b\}$. Figure 3 shows the corresponding weighted transducer. $\hat{T}(x)$ is finite for all $x \in \Sigma^*$, the feature space F associated to K has then dimension 2 and the points mapped by the corresponding feature mapping are those with non-negative integer coordinates. Let the sample include all non-empty sequences, $\mathcal{S} = \Sigma^+$, and let H be the hyperplane going through the point $(0, 0)$ with a positive irrational slope α. By definition, H does not cross any point with positive integer coordinates (p, q), since $\frac{p}{q} \in \mathbb{Q}$, thus it is indeed a separating hyperplane for $\mathcal{S}$. But, since $\mathbb{Q}$ is dense in $\mathbb{R}$, for any $\epsilon > 0$, there exists a rational number $\frac{p}{q}$ such that $|\frac{p}{q} - \alpha| < \epsilon$. This shows that there are points with positive integer coordinates arbitrarily close to H and thus that the margin associated to H is zero.

The language separated by H is the non-regular language of non-empty sequences with α times more bs than as:[6]

$$L = \{x \in \Sigma^+ : |x|_b > \alpha |x|_a\}. \tag{19}$$

The relationship between the existence of a positive margin for a PDS rational kernel and an isolated cut-off point is not straightforward. By Theorem 2, if for all $x \in \Sigma^+, S(x) > \rho > 0$, then there exists a probabilistic automaton B with N states such that $\forall x \in \Sigma^+,\ |B(x) - \frac{1}{N}| > \frac{\rho}{c^{|x|}}$. But, since $|x|$ can be arbitrarily large, this does not guarantee an isolated cut-point.

4.3 Positive Margin

When the values $T(x, y)$ taken by the transducer T for all pairs of sequences (x, y) are integers within a finite range $[0, N]$, then linear separation with a PDS rational kernel defined by $K = T \circ T^{-1}$ guarantees a positive margin. The feature mapping Φ associated to K then also takes integer values in $[0, N]$.

Proposition 1. *Let C be a class of concepts defined over a set X that is linearly separable using a mapping $\Phi : X \to \{0, 1, \ldots, N\}^{\mathbb{N}}$ 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, span(w). Thus, $\Phi'(x)$ is a finite-dimensional vector for any $x \in X$ with discrete coordinates in $\{0, 1, \ldots, N\}$. Thus, the set of $\mathcal{S} = \{\Phi'(x) : x \in X\}$ is finite. Since for any $x \in X$, $\langle w, \Phi(x)\rangle = \langle w, \Phi'(x)\rangle$, the margin can be defined over a finite set:

$$\rho = \inf_{x \in X} \frac{|\langle w, \Phi'(x) + b|\rangle}{\|w\|} = \min_{z \in \mathcal{S}} \frac{|\langle w, z\rangle + b|}{\|w\|}, \tag{20}$$

which implies $\rho > 0$ since $|\langle w, z\rangle + b| > 0$ for all $z \in \mathcal{S}$. □

Many of the PDS rational kernels used in practice follow these conditions. In particular, for kernels such as the subsequence kernels, the transducer T takes only values 0 or 1.

When the existence of a positive margin is guaranteed as in the case of rational kernels with finite range, the following theorem applies.

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

$$\mathcal{F} = \{x \mapsto \langle w, \Phi(x)\rangle : \|w\| \leq 1, \|\Phi(x)\| \leq R\}. \tag{21}$$

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

$$\frac{\alpha_0}{m}\left(\frac{R^2}{\rho_0^2}\log^2 m + \log(\frac{1}{\delta})\right). \tag{22}$$

[6] When α is a rational number, it can be shown that the margin is positive, the language L being still non-regular.

Proof. Fix a concept $c \in C$. By assumption, c is finitely linearly separable 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 a standard margin bound of Bartlett and Shawe-Taylor [1]. □

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.

5 Algorithm for Finite Range Rational Kernels

The previous section showed that PDS rational kernels with finite feature values ensure a positive margin and thus learning with the margin-based guarantees previously described.[7] However, while it is natural and often straightforward to come up with a transducer mapping input sequences to the features sequences, that transducer often cannot be readily used for the definition of the kernel. This is because it may contain several paths with the same output feature sequence and the same input sequence.

For example, it is easy to come up with a transducer mapping each string to the set of its subsequences. Figure 5(a) shows a simple one-state transducer doing that. But, when applied to the sequence $x = aba$, that transducer generates two paths with input x and output a because a appears twice in x. Instead, we need to construct a transducer that contains exactly one path with input x and output a. Figure 5(b) shows a subsequence kernel with that property.

The construction of such a transducer is not trivial even for this simple case. One may then ask if there exists a general procedure for constructing a transducer with multiplicity one from a given transducer. This section describes a novel and general algorithm that serves precisely that purpose.

The algorithm takes as input an arbitrary (unweighted) transducer T and outputs a transducer T' that is unambiguous in the following way: for any pair of input and output sequence (x, y) labeling a successful path of T, T' contains exactly one successful path with that label. We will refer to our algorithm as the *double-tape disambiguation*. Note that our algorithm is distinct from the standard *disambiguation algorithm* for transducers [2] which applies only to transducers that represent a partial function mapping input sequences to output sequences and which generates a transducer with unambiguous input.

To present the algorithm, we need to introduce some standard concepts of word combinatorics [17]. To any $x \in \Sigma^*$, we associate a new element x' denoted by x^{-1} and extend string concatenation so that $xx' = x'x = \epsilon$. We denote by $(\Sigma^*)^{-1}$ the set of all these new elements. The *free group generated by* Σ denoted by $\Sigma^{(*)}$ is the set of all elements that can be written as a concatenation of elements of Σ^* and ${\Sigma^*}^{-1}$. We say that an $x \in \Sigma^{(*)}$ of the free group is *pure* if $x \in \Sigma^* \cup {\Sigma^*}^{-1}$ and we denote that set by $\Pi = \Sigma^* \cup {\Sigma^*}^{-1}$.

The algorithm constructs a transducer T' whose states are pairs (p, m) where $p \in Q$ is a state of the original transducer and a m is a multiset of triplets (q, x, y) with $q \in Q$ and

[7] This section concentrates on kernels with just binary feature values but much of our analysis generalizes to the more general case of finite feature value.

$x, y \in \Sigma^* \cup (\Sigma^*)^{-1}$. Each triplet (q, x, y) indicates that state q can be reached from the initial state by reading either the same input string or the same output string as what was used to reach p. x and y serve to keep track of the extra or missing suffix of the labels of the path leading to q versus the current one used to reach p.

Let (u, v) denote the input and output label of the path followed to reach p, and (u', v') the labels of the path reaching q. Then, x and y are defined by:

$$x = (u)^{-1}u' \qquad y = (v)^{-1}v'. \tag{23}$$

We define a partial transition function δ for triplets. For any (q, x, y) and $(a, b) \in (\Sigma \cup \{\epsilon\})^2 - \{(\epsilon, \epsilon)\}$, $\delta((q, x, y), a, b))$ is a multiset containing

$$(q', xa^{-1}a', yb^{-1}b'), \tag{24}$$

if $(q, a', b', q') \in E, xa^{-1}a \in \Pi$, and $yb^{-1}b' \in \Pi$, $\delta((q, x, y), a, b)) = \emptyset$ otherwise. We further extend δ to multisets by defining $\delta(m, a, b))$ as the multiset of all $\delta((q, x, y), a, b))$ with (q, x, y) in m.

The set of initial states I' of T' are the states $(i, (i, \epsilon, \epsilon))$ with $i \in I$. Starting from an initial state, the algorithm creates new transitions of T' as follows. When $p, a, b, p' \in E$ and that it does not generate ambiguities (as we shall see later), it creates a transition from state (p, m) to state $(p', \delta(m))$ with input label a and output label b.

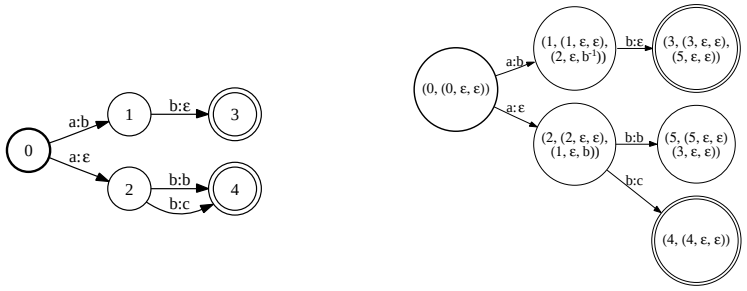
At the price of splitting final states, without loss of generality, we can assume that the transducer T does not admit two paths with the same labels leading to the same final state. When there are k paths in T with the same input and output labels and leading to distinct final states $p_1, \ldots, p_k$, the algorithm must disallow all but one in T'. Observe that these paths correspond to paths in T' ending at the states (p_i, m), $i \in [1, k]$, with the same multiset m, which therefore contains (ϵ, ϵ) with multiplicity k. To guarantee the result to be unambiguous, the algorithm allows only one of the states (p_i, m), $i \in [1, k]$ to be final. This preserves the mapping defined by T since it does not affect other paths leaving (p_i, m), $i \in [1, k]$. The choice of the particular state to keep final is arbitrary and does not affect the result. Different choices lead to transducers with different topologies that are all equivalent.

The algorithm described thus far can be applied to acyclic transducers since it creates at most a finite number of states in that case and since it disallows ambiguities. Figure 4 illustrates the application of the algorithm in a simple case. In the general case, to avoid the creation of infinitely many states, after disambiguation, a transition of T' with labels (a, b) and destination state (p, m') with $m \subset m'$ is directed instead to (p, m), if (p, m) admits an incoming transition labeled with (a, b).

The following gives the pseudocode of our algorithm. The algorithm takes as input T and outputs the transducer $T' = (\Sigma, \Sigma, Q', I', F', E')$. The algorithm uses a queue S containing the set of states of T' yet to be examined. The queue discipline of S can be arbitrarily chosen. Each time through the loop of lines 5-16 a new state (p, m) is extracted from S. For each of its outgoing transitions, a new transition e' is created when this does not yield ambiguity (condition of line 13). An existing destination state is selected for e' (line 10) to avoid creating unnecessary states.

DOUBLE-TAPE-DISAMBIGUATION(T)

1 $S \leftarrow Q' \leftarrow I' \leftarrow \{(i, (\epsilon, \epsilon)) : i \in I\}$



(a) (b)

Fig. 4. Illustration of the application of the double-tape disambiguation algorithm. (a) Transducer T. (b) Equivalent double-tape unambiguous transducer T' obtained by application of the algorithm. The destination state of the transition labeled with $b{:}b$ is made non-final by the algorithm, which makes the result unambiguous. That state is non-coaccessible and can be later removed by a standard trimming algorithm for automata and transducers.

```
2  if {(i, (ϵ, ϵ)) : i ∈ I ∩ F} ≠ ∅
3     then F' ← {(i_0, (ϵ, ϵ))} ▷ selected arbitrarily from {(i, (ϵ, ϵ)) : i ∈ I ∩ F}
4     else F ← ∅
5  while S ≠ ∅
6        do (p, m) ← head(S); DEQUEUE(S)
7           for each (p, a, b, p') ∈ E
8              do m' ← δ(m, a, b)
9                 if p' ∉ F or ∄(p'', m') ∈ F' and
                     ∃((p_0, m_0), a, b, (p_1, m_1)) ∈ E' with m_1 ⊂ m'
10                      then e' ← ((p, m), a, b, (p_1, m_1))
11                      else e' ← ((p, m), a, b, (p', m'))
12                           Q' ← Q' ∪ {(p', m')}
13                           if p' ∈ F and ∄(p'', m') ∈ F'
14                              then F' ← F' ∪ {(p', m')}
15                   E' ← E' ∪ {e'}
16 return T' = (Σ, Σ, Q', I', F', E')
```

Theorem 5. *For any transducer T, the algorithm* Double-Tape-Disambiguation *produces an equivalent transducer T' that is double-tape unambiguous.*

Proof. We give a sketch of the proof. By definition of the test of line 9, the output transducer T' is double-tape unambiguous. The equivalence and termination of the algorithm are clear for acyclic input transducers since the destination state of e' is (p', m') and left unchanged. In the general case, an argument based on the context of the states (p_1, m_1) and (p', m') and a theorem of Eilenberg [10] for unambiguous rational sets shows that selecting (p', m') as a destination state does not change the mapping defined by T and thus that T' is equivalent to T. □

The input transducer T can be determinized, viewed as an acceptor defined over pairs of input-output symbols. When it is deterministic, then the transducer T' output by the algorithm is also necessarily deterministic, by construction. The application of the standard automata minimization can thus help reduce the size of T'.

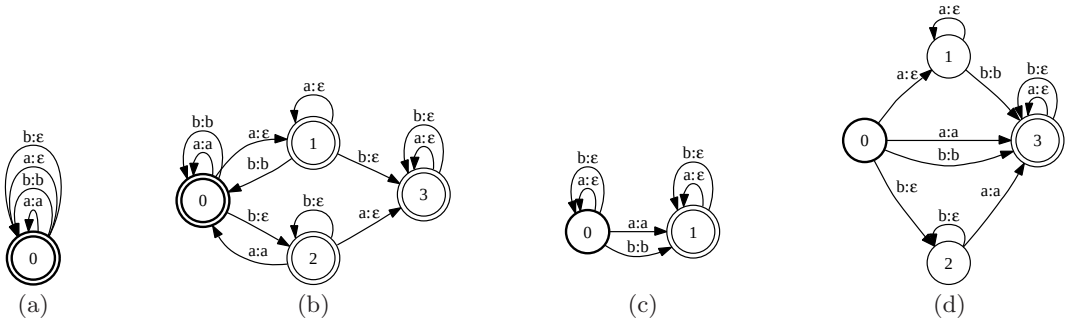


Fig. 5. Applications of the double-tape disambiguation algorithm. (a) Transducer T_0 associating to each input string $x \in \Sigma^*$ the set of its subsequences (with multiplicity) for $\Sigma = \{a, b\}$. (b) Subsequence transducer T associating to each string $x \in \Sigma^*$ the set of its subsequences with multiplicity one regardless of the number of occurrences of the subsequences in x. Unigram transducers for $\Sigma = \{a, b\}$. (c) Transducer T_0 associating to each input string $x \in \Sigma^*$ the set of its unigrams a and b (with multiplicity). (d) Unigram transducer T associating to each string its unigrams with multiplicity one.

Figures 5(a)-(b) and Figures 5(c)-(d) show examples of applications of our algorithm to some kernels of interest after minimization. Figure 5(b) shows the subsequence transducer resulting from the application of our algorithm to the transducer of Figure 5(a) which counts subsequences with their multiplicity. Figure 5(b) shows the subsequence transducer obtained by applying our algorithm to the transducer of Figure 5(a) which counts unigrams with their multiplicity. In both cases, the resulting transducers are non-trivial and not straightforward to define even for such relatively simple examples. The double-tape disambiguation algorithm can be used as a tool to define finite range rational kernels based on such transducers.

6 Conclusion

We presented a general study of learning and linear separability with rational kernels, the sequence kernels commonly used in computational biology and natural language processing. We gave a characterization of the family of languages linearly separable with rational kernels demonstrating the central role of stochastic languages in this setting. We also pointed out several important properties of languages separable with a fixed rational kernel in terms of finite Boolean combination of languages.

Rational kernels with finite range stand out as a particularly interesting family of kernels since they verify this property and guarantee a positive margin. The double-tape disambiguation algorithm we presented can be used to create efficiently such kernels from a finite-state transducer defining the mapping to feature sequences. The algorithm is of independent interest for a variety of other applications in text and speech processing where such a disambiguation is beneficial.

References

1. 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.
2. Jean Berstel. *Transductions and Context-Free Languages.* Teubner Studienbucher: Stuttgart, 1979.

3. 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.
4. Michael Collins and Nigel Duffy. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.
5. Corinna Cortes, Patrick Haffner, and Mehryar Mohri. Rational Kernels: Theory and Algorithms. *Journal of Machine Learning Research (JMLR)*, 5:1035–1062, 2004.
6. Corinna Cortes and Mehryar Mohri. Moment Kernels for Regular Distributions. *Machine Learning*, 60(1-3):117–134, September 2005.
7. Corinna Cortes, Mehryar Mohri, Ashish Rastogi, and Michael Riley. Efficient Computation of the Relative Entropy of Probabilistic Automata. In *Proceedings of the 7th Latin American Symposium (LATIN 2006)*, volume 3887 of *Lecture Notes in Computer Science*, pages 323–336, Valdivia, Chile, March 2006. Springer-Verlag, Heidelberg, Germany.
8. Corinna Cortes and Vladimir N. Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995.
9. Francois Denis and Yann Esposito. Rational stochastic languages. In *Proceedings of The 16th Annual Conference on Computational Learning Theory (COLT 2006)*, Lecture Notes in Computer Science. Springer, Heidelberg, Germany, 2006.
10. Samuel Eilenberg. *Automata, Languages and Machines*, volume A–B. Academic Press, 1974–1976.
11. David Haussler. Convolution Kernels on Discrete Structures. Technical Report UCSC-CRL-99-10, University of California at Santa Cruz, 1999.
12. Leonid Kontorovich, Corinna Cortes, and Mehryar Mohri. Kernel Methods for Learning Languages. *Theoretical Computer Science*, (submitted), 2006.
13. 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.
14. Werner Kuich and Arto Salomaa. *Semirings, Automata, Languages*. Number 5 in EATCS Monographs on Theoretical Computer Science. Springer-Verlag, 1986.
15. Christina Leslie, Eleazar Eskin, Jason Weston, and William Stafford Noble. Mismatch String Kernels for SVM Protein Classification. In *NIPS 2002*, Vancouver, Canada, March 2003. MIT Press.
16. 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.
17. M. Lothaire. *Combinatorics on Words*, volume 17 of *Encyclopedia of Mathematics and Its Applications*. Addison-Wesley, 1983.
18. Azaria Paz. *Introduction to probabilistic automata*. Academic Press, New York, 1971.
19. Michael O. Rabin. Probabilistic automata. *Information and Control*, 6:230–245, 1963.
20. Arto Salomaa and Matti Soittola. *Automata-Theoretic Aspects of Formal Power Series*. Springer-Verlag, 1978.
21. Imre Simon. Piecewise testable events. In *Automata Theory and Formal Languages*, pages 214–222, 1975.
22. Paavo Turakainen. Generalized Automata and Stochastic Languages. *Proceedings of the American Mathematical Society*, 21(2):303–309, May 1969.
23. Vladimir N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, 1998.
24. A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K.-R. Müller. Engineering support vector machine kernels that recognize translation initiation sites. *Bioinformatics*, 16(9):799–807, 2000.