
Machine Learning, 60, 117–134, 2005

Moment Kernels for Regular Distributions

CORINNA CORTES corinna@google.com
Google Labs, 1440 Broadway, New York, NY 10018

MEHRYAR MOHRI mohri@research.att.com
AT&T Labs—Research, 180 Park Avenue, Florham Park, NJ 07932

Editors: Dan Roth and Pascale Fung

Abstract. Many machine learning problems in natural language processing, transaction-log analysis, or computational biology, require the analysis of variable-length sequences, or, more generally, distributions of variable-length sequences.

Kernel methods introduced for fixed-size vectors have proven very successful in a variety of machine learning tasks. We recently introduced a new and general kernel framework, *rational kernels*, to extend these methods to the analysis of variable-length sequences or more generally distributions given by weighted automata. These kernels are efficient to compute and have been successfully used in applications such as spoken-dialog classification with Support Vector Machines.

However, the rational kernels previously introduced in these applications do not fully encompass distributions over alternate sequences. They are based only on the counts of co-occurring subsequences averaged over the alternate paths without taking into accounts information about the higher-order moments of the distributions of these counts.

In this paper, we introduce a new family of rational kernels, *moment kernels*, that precisely exploits this additional information. These kernels are distribution kernels based on moments of counts of strings. We describe efficient algorithms to compute moment kernels and apply them to several difficult spoken-dialog classification tasks. Our experiments show that using the second moment of the counts of n-gram sequences consistently improves the classification accuracy in these tasks.

Keywords: statistical learning, kernel methods, rational kernels, string kernels, weighted automata, weighted finite-state transducers, spoken-dialog classification

1. Introduction

Many machine learning problems in natural language processing require the analysis of variable-length sequences. These may be sequences of words, or phonemes, or other linguistic units possibly augmented with some tags or parentheses as in part-of-speech tagging or parsing. Similarly, in bioinformatics, the analysis of protein sequences or other biological sequences is required. Transaction-logs such as those recording visits to a web site also form variable-length sessions that can be analyzed to cluster customers or otherwise derive information about the behavior of the visitors or the functionality of the site.

More generally, in many applications, the analysis of distributions over variable-length sequences is needed. Indeed, the output of complex systems combining multiple knowledge sources, e.g., a large-vocabulary speech recognition or an information

extraction system is typically a weighted automaton compactly representing a large set of alternative sequences or paths, where the weights rank different hypotheses according to the underlying models of these systems and represent the probability of correctness of the sequences.

A common approach in statistical learning techniques such as Support Vector Machines (SVMs) (Boser, Guyon, & Vapnik, 1992; Cortes & Vapnik, 1995; Vapnik, 1998) is that of kernel methods (Schölkopf & Smola, 2002) which allow an implicit and efficient computation of dot products in a high-dimensional feature space. However, these kernels have been primarily developed for fixed-sized vectors. We recently introduced a general kernel framework based on weighted transducers, *rational kernels*, to extend kernel methods to the analysis of variable-length sequences or more generally weighted automata (Cortes, Haffner, & Mohri, 2003a; 2003b).

This framework includes many of the string kernels introduced for computational biology or text classification applications (Cortes, Haffner, & Mohri, 2003a), e.g., David Haussler's convolution kernels for strings (1999), kernels discussed by Watkins (1999), the paths kernels of Takimoto and Warmuth (2003), other string kernels applied to bioinformatics problems (Leslie et al., 2003), or the gappy kernels applied to text classification problems by Lodhi et al. (2001).[1] There exists a general and efficient algorithm for computing rational kernels, which can be used to compute all of the kernels just mentioned (Cortes, Haffner, & Mohri, 2003b).

We also previously introduced a family of rational kernels measuring the similarity between the weighted automata output by speech recognition systems and showed that they can be successfully used for spoken-dialog classification with SVMs (Cortes, Haffner, & Mohri, 2003b). However, these kernels do not fully encompass distributions over alternate sequences, they are based only on the expected counts of co-occurring subsequences aggregated over the alternate paths and ignore information about higher order moments of the distributions of these counts.[2]

This paper introduces a new family of rational kernels, *moment kernels*, that precisely exploits this additional information and generalizes the kernels we had previously defined. Moment kernels are distribution kernels based on higher moments of counts of strings. We describe efficient algorithms to compute moment kernels and apply them to several difficult spoken-dialog classification tasks. Our experiments show that using the second moment of the counts of n-gram sequences consistently improves the classification accuracy in these tasks.

The paper is organized as follows. It first briefly presents the essential concepts and algorithms related to rational kernels. It then introduces the family of moment kernels, which are kernels based on the moments of the counts of n-gram sequences appearing in weighted automata. It describes in detail new algorithms for computing the moments of the count of an arbitrary string in a weighted automaton and gives the proof of their correctness. This shows that moment kernels are rational kernels and that they generalize those based on just the expected counts. The last section details the application of moment kernels to spoken-dialog classification and reports the results of our experiments in several difficult tasks.

2. Rational kernels

This section gives an overview of rational kernels, their definition, and their properties.

2.1. Weighted automata and transducers

We start with a brief presentation of some of the basic notions and algorithms related to weighted automata and finite-state transducers that are needed for the introduction of rational kernels.

A *weighted finite-state transducer* T is an automaton in which each transition, in addition to its usual input label, is augmented with an output label from a possibly different alphabet, and carries some weight. It is thus 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 where ϵ represents the empty string, $\lambda : I \to \mathbb{R}$ the initial weight function, and $\rho : F \to \mathbb{R}$ the final weight function mapping F to $\mathbb{R}$. In what follows, the weights can be interpreted as probabilities, thus they are multiplied along a path.

Let $R(I, x, y, F)$ denote the set of paths from an initial state $p \in I$ to a final state $q \in F$ with input label x and output label y, $w[\pi]$ the weight of path π, $\lambda[p[\pi]]$ the initial weight of the origin state of π, and $\rho[n[\pi]]$ the final weight of its destination. A transducer T is *regulated* if the output weight associated by T to any pair of strings $(x, y) \in \Sigma^* \times \Delta^*$:

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

is well-defined and in $\mathbb{R}$. A *weighted automaton* $A = (\Sigma, Q, I, F, E, \lambda, \rho)$ is defined in a similar way by simply omitting the output (or input) labels. A is *regulated* if the weight associated by A to any string $x \in \Sigma^*$:

$$[\![A]\!](x) = \sum_{\pi \in R(I,x,F)} \lambda[p[\pi]] \cdot w[\pi] \cdot \rho[n[\pi]] \tag{2}$$

is well-defined and in $\mathbb{R}$, where $R(I, x, F)$ denotes the set of paths labeled with x from an initial state to a final state. We denote by $\Pi_2(T)$ (Π_1) the weighted automaton obtained from T by removing its input labels (resp. output labels).

A general *composition* operation similar to the composition of relations can be defined for weighted finite-state transducers when the output alphabet of one transducer matches the input alphabet of another transducer. The composition of two weighted finite-state transducers $T_1 = (\Sigma, \Omega, Q_1, I_1, F_1, E_1, \lambda_1, \rho_1)$ and $T_2 = (\Omega, \Delta, Q_2, I_2, F_2, E_2, \lambda_2, \rho_2)$ is a weighted transducer denoted by $T_1 \circ T_2$ and defined by:

$$[\![T_1 \circ T_2]\!](x, y) = \sum_{z \in \Omega^*} [\![T_1]\!](x, z) \cdot [\![T_2]\!](z, y) \tag{3}$$

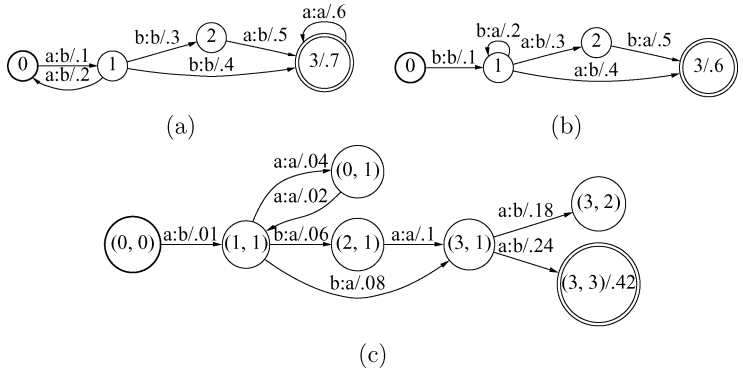

Figure 1. (a) Weighted transducer T_1. (b) Weighted transducer T_2. (c) Composition of T_1 and T_2. Initial states are represented by bold circles, final states by double circles. Inside each circle, the number indicates the state number. For final states, the value of the final weight function ρ is indicated after the slash symbol. Arrows represent transitions. Each transition is labeled with an input label and an output label separated by a colon, and a transition carries some weight indicated after the slash symbol, e.g., the transition from state 1 to state 2 in the first transducer has input label a, output label b, and weight .1.

There exists an algorithm for computing and constructing $T = T_1 \circ T_2$ from T_1 and T_2 (Pereira & Riley, 1997; Mohri, Pereira, & Riley, 1996). Its complexity is quadratic, $O(|T_1||T_2|)$, where $|T_i|$, $i = 1, 2$, is the size of T_i, that is the sum of the number of states and transitions of T_i. The states of T are identified as pairs of a state of T_1 and a state of T_2. A state (q_1, q_2) in $T_1 \circ T_2$ is an initial (final) state if and only if q_1 is an initial (resp. final) state of T_1 and q_2 is an initial (resp. final) state of T_2. The transitions of T are the result of matching a transition of T_1 from state q_1 to state q_1' and a transition of T_2 from state q_2 to state q_2' as follows: (q_1, a, b, w_1, q_1') and (q_2, b, c, w_2, q_2') produce the transition $((q_1, q_2), a, c, w_1 \cdot w_2, (q_1', q_2'))$ in T.[3] Figure 1(c) shows the result of the application of the algorithm to the weighted transducers of Figure 1(a) and (b).

The definition of composition extends naturally to weighted automata since a weighted automaton can be viewed as a weighted transducer with identical input and output labels for each transition. The corresponding transducer associates $[\![A]\!](x)$ to all pairs (x, x), and 0 to all other pairs. Thus, the composition of a weighted automaton A_1 and a weighted transducer T_2 is simply defined by:

$$[\![A_1 \circ T_2]\!](x, y) = \sum_{x \in \Omega^*} [\![A_1]\!](x) \cdot [\![T_2]\!](x, y) \tag{4}$$

and can be computed as previously described.

2.2. *Definition of rational kernels*

A kernel K is a function mapping pairs of strings in $\Sigma^* \times \Sigma^*$ to the real numbers, $\mathbb{R}$. K is said to be *rational* if there exists a weighted transducer $T = (\Sigma, \Sigma, Q, I, F, E, \lambda, \rho)$ such that for all $x, y \in \Sigma^*$:[4]

$$K(x, y) = [\![T]\!](x, y) \tag{5}$$

We will refer to T as the *weighted transducer associated to the rational kernel* K. Rational kernels can be naturally extended to kernels over weighted automata. Let A and B be two weighted automata, then $K(A, B)$ is defined by:

$$K(A, B) = \sum_{(x,y)\in\Sigma^*\times\Sigma^*} [\![A]\!](x) \cdot [\![T]\!](x, y) \cdot [\![B]\!](y) \tag{6}$$

for all weighted automata A and B such that the sum:

$$\sum_{(x,y)\in\Sigma^*\times\Sigma^*} [\![A]\!](x) \cdot [\![T]\!](x, y) \cdot [\![B]\!](y)$$

is well-defined and in $\mathbb{R}$. This sum is always defined and in $\mathbb{R}$ when A and B are acyclic weighted automata since the sum then runs over a finite set. It is also well-defined for all A, B, and T representing probability distributions. In view of the definition of composition given earlier, when $K(A, B)$ is defined, Eq. (6) can be rewritten as:

$$K(A, B) = \sum_{(x,y)\in\Sigma^*\times\Sigma^*} [\![A \circ T \circ B]\!](x, y) \tag{7}$$

Thus, $K(A, B)$ is simply the sum of the weights of all the paths of $A \circ T \circ B$ from an initial state to a final state. $A \circ T \circ B$ can be computed using the composition algorithm previously described, and a shortest-distance algorithm or a forward-backward algorithm can be used to compute the sum of the weights of all its paths.

2.3. Properties of rational kernels

To ensure the convergence of the training algorithm to a unique optimum when using learning techniques such as SVMs, the kernels used must be *positive definite symmetric* (PDS), or, equivalently, verify Mercer's condition. A kernel K is PDS if it is symmetric ($K(x, y) = K(y, x)$ for all $x, y \in \Sigma^*$) and the matrix $K(x_i, x_j)_{i,j\leq n}$ for all $n \geq 1$ and all $\{x_1, \ldots, x_n\} \subseteq X$ has only non-negative eigenvalues. This condition guarantees the existence of a Hilbert space and a dot product associated to the kernel considered.

There exists a simple and general method for constructing a PDS rational kernel from an arbitrary weighted transducer T (Cortes, Haffner, & Mohri, 2003a). We denote by T^{-1} the *inverse of* T, that is the transducer obtained from T by swapping input and output labels of each transition. When the transducer $T \circ T^{-1}$ is regulated, the rational kernel associated to $T \circ T^{-1}$ is PDS. This can be illustrated with a simple example. As we shall see later, there exists a transducer T that can compute $|x|_z$ the number of occurrences of a given string $z \in \Sigma^*$ in a sequence x, $[\![T]\!](x, z) = |x|_z$. Thus, $T \circ T^{-1}$ defines a PDS rational kernel K based on the product of the number of occurrences of z between two sequences. Figure 2 shows the $T \circ T^{-1}$ transducer associated with the rational kernel based on the common counts of the sequence ba when the alphabet is $\Sigma = \{a, b\}$.

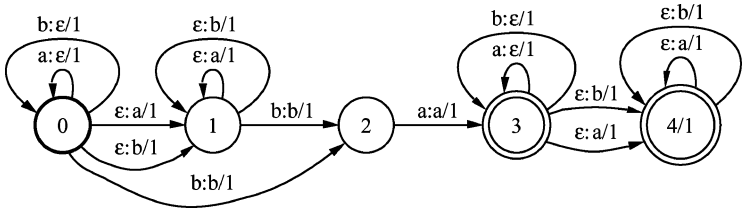

Figure 2. Rational kernel based on common counts of the sequence *ba* over the alphabet $\Sigma = \{a, b\}$. The states 0 and 1 disregard arbitrary prefixes of the sequences to be matched, the states 2 and 3 perform the actual match, and the states 3 and 4 disregard suffixes. Each successful path contributes a weight of 1, thus the sum of all the successful paths is indeed the sum of common occurrences of the sequence *ba*.

An important property of PDS rational kernels as defined above is that they are closed under sum, product, and Kleene-Closure (Cortes, Haffner, & Mohri, 2003a). Furthermore, the transducer associated to the sum of two PDS rational kernels can simply be obtained by taking the sum of the weighted transducers associated to these kernels. This can be used to construct more complex PDS rational kernels from simpler ones.

3. Kernels based on counts

A kernel can be viewed as a similarity measure. Many of the kernels introduced for strings in computational biology or natural language processing applications are based of the following idea: two strings are similar when they share many common factors.[5] Thus, kernels for strings may be defined for example as the sum of the products of all common factors between two sequences.

To generalize this idea to kernels defined over weighted automata, the distribution of the counts of sequences in each automaton must be taken into account. This section gives the definition of these moments and defines a family of kernels based on these moments.

3.1. Definition

Let $A = (\Sigma, Q, I, F, E, \lambda, \rho)$ be an arbitrary weighted automaton. We are interested in *counting* the occurrences of a sequence x in A while taking into account the weight of the paths where they appear. When A is *stochastic*, i.e., when it is deterministic and the sum of the weights of the transitions leaving any state is 1, it can be viewed as a probability distribution P over all strings Σ^*. The weight $[\![A]\!](u)$ associated by A to a string $u \in \Sigma^*$ is then $P(u)$. This leads to the following definition:

Let m be a positive integer. Let $c_m^A(x)$ denote the *m-th moment of the count* of the sequence x in A defined by:

$$c_m^A(x) = \sum_{u \in \Sigma^*} |u|_x^m \, [\![A]\!](u) \tag{8}$$

where $|u|_x$ denotes the number of occurrences of x in the string u labeling one or several successful paths of A. We define the *m-th* moment of the count of x as above regardless of whether A is stochastic or not. In many applications, the weighted automaton A is acyclic, e.g., it is the output of a speech recognition system, but our algorithms are general and do not assume A to be acyclic.

In previous work (Cortes, Haffner, & Mohri, 2003b, 2003c), we defined a family of kernels K_1^n based on the expected counts of common n-gram sequences between two automata A_1 and A_2:

$$K_1^n(A_1, A_2) = \sum_{|x|=n} c_1^{A_1}(x) c_1^{A_2}(x) \tag{9}$$

where $c_1^{A_i}(x)$, $i = 1, 2$, denotes the expected count of x in the weighted automaton A_i. With these kernels, two automata are viewed as similar when they share common n-gram subsequences with high expected counts. In many cases, one may wish to consider not just the n-gram sequences of a particular order to measure the similarity between two weighted automata but all n-grams up to a given order. The resulting kernel is also a PDS rational kernel since the sum a finite number of PDS rational kernels is a PDS rational kernel. Thus, we define an *n-gram rational kernel* $\mathcal{K}_1^n$ as the PDS rational kernel obtained by taking the sum of all K_1^j, with $1 \leq j \leq n$:

$$\mathcal{K}_1^n = \sum_{j=1}^{n} K_1^j \tag{10}$$

These kernels can be generalized to take into account other moments of the distributions of the counts of n-gram sequences. We define a general family of kernels, denoted by K_m^n, $m, n \geq 1$ and defined by:

$$K_m^n(A_1, A_2) = \sum_{|x|=n} c_m^{A_1}(x) c_m^{A_2}(x) \tag{11}$$

In Section 4 we show that there are weighted transducers T_m^n that can compute $c_m^{A_1}(x)$ for all n-gram sequence x. Thus, these kernels are rational kernels and their associated transducers are $T_m^n \circ T_m^{n-1}$:

$$K_m^n(A_1, A_2) = \sum_{x,y} [\![A_1 \circ \left(T_m^n \circ T_m^{n-1}\right) \circ A_2]\!](x, y) \tag{12}$$

Since their definition is based on weighted transducers of the type $T \circ T^{-1}$, they are PDS kernels (Cortes, Haffner, & Mohri, 2003a). This guarantees the convergence of training for discriminant classification algorithms such as SVMs when using these kernels. As in the case of expected counts, one may wish to take into account all n-gram sequences up to a given order. Since a finite sum of PDS rational kernels is PDS, we define an n-gram

moment kernel $\mathcal{K}_m^n$ as the PDS rational kernel obtained by taking the sum of all K_m^j, with $1 \leq j \leq n$:

$$\mathcal{K}_m^n = \sum_{j=1}^{n} K_m^j \tag{13}$$

Moment kernels can be used to measure the similarity of two weighted automata A_1 and A_2 based on the higher order moments of the counts of their common n-gram sequences. Since rational kernels are closed under rational operations (Cortes, Haffner, & Mohri, 2003a), moment kernels can be combined using sum, product, or the Kleene-Closure to create more complex rational kernels taking advantage of higher order moments of the counts of subsequences. They can also be combined using other functions to define useful positive definite kernels. We shall later reports the results of our preliminary experiments in several difficult spoken-dialog classification tasks using moment kernels in combination with n-gram kernels.

4. Algorithms

This section describes efficient algorithms for computing the moments of the distributions of the counts of an arbitrary sequence x appearing in a weighted automaton A and gives the proof of the correctness of the algorithms. It also presents algorithms for computing the moments of the counts of all sequences $x \in L(X)$, where $L(X)$ is the language described by a regular expression X.

Our algorithms for computing the moments of the count of a sequence x are based on the definition of suitable weighted transducers. We start with the simpler case where the sequence x is *aperiodic*.

A positive integer p is said to be a *period* of a string $x = a_1 a_2 \cdots a_n$ if $a_i = a_{i+p}$ for $i = 1, \ldots, n - p$. Note that $|x|$, the length of x, is always a period of x. The smallest period of x is called *the period* of x. x is said to be *aperiodic* if its period coincides with its length $|x|$. The period of the string *ba* is 2, since *ab* is the shortest repeated pattern in that string. The string *abb* is aperiodic since it contains no repeated pattern shorter than itself. When a string x is aperiodic, two consecutive occurrences of x cannot overlap. This may happen in the case of non-aperiodic strings, e.g., *ababa* contains two overlapping occurrences of *aba*.

We will denote by $\Omega \subset \Sigma^*$ the set of aperiodic strings over the alphabet Σ.

4.1. Case of aperiodic sequences

Proposition 1. *Let $x \in \Omega$. Then, for any positive integer m, there exists a weighted transducer T_m such that for any weighted automaton A:*

$$[\![\Pi_2(A \circ T_m)]\!](x) = c_m^A(x) \tag{14}$$

Proof: Let m be a positive integer. Let α_m be the weighted regular expression (or rational power series) defined by:

$$\alpha_m = \sum_{k=0}^{m} k!\, S(m,k)\, (\Sigma^* x)^k \Sigma^* \tag{15}$$

where $S(m,k)$, $k = 1, \ldots, m$, denote the Stirling numbers of the second kind (van Lint & Wilson, 1992). $S(m,k)$ represents the number of possible partitions of an m-set into k non-empty subsets. The Stirling numbers of the second type verify the following identity:

$$\sum_{k=0}^{m} \binom{N}{k} k!\, S(m,k) = N^m \tag{16}$$

The weight associated by a weighted regular expression X to a string u is denoted by (X, u). The weight associated by α_m to a string $u \in \Sigma^*$ is thus:

$$(\alpha_m, u) = \sum_{k=0}^{m} k!\, S(m,k)\, ((\Sigma^* x)^k \Sigma^*, u) \tag{17}$$

The number $((\Sigma^* x)^k \Sigma^*, u)$ corresponds to the number of occurrences of $x\Sigma^* x \Sigma^* \cdots \Sigma^* x$ in u (with k repetitions of x). Since x is aperiodic, two consecutive occurrences of x cannot overlap. Thus, this is the number of ways of choosing k positions of x in u. Let $N = |u|_x$ denote the number of occurrences of x in u. Then, $((\Sigma^* x)^k \Sigma^*, u) = \binom{N}{k}$. Thus,

$$(\alpha_m, u) = \sum_{k=0}^{m} \binom{N}{k} k!\, S(m,k) = N^m \tag{18}$$

Since α_m is a weighted regular expression, by the theorem of Kleene, Schützenberger (1961), there exists a weighted automaton A_m realizing α_m. Let T_m be a weighted transducer with input automaton A_m and with output reduced to x. T_m verifies exactly the hypotheses of the proposition. □

The next proposition shows that there exists an efficient algorithm for computing the moments of the counts of an aperiodic sequence x.

Proposition 2. *Let $x \in \Omega$. There exists an algorithm for computing $c_m^A(x)$ for any weighted automaton A in $O(m|A||x|)$ time and $O(m|A||x|)$ space.*

Proof: By Proposition 1, given the complexity of composition, $c_m^A(x)$ can be computed in $O(|A||T_m|)$ time and space where T_m is a weighted transducer with input an automaton representing α_m and with output x. It is clear that there exists a weighted transducer with only $m|x| + 1$ states realizing this mapping. Figure 3 shows that transducer. Clearly, its

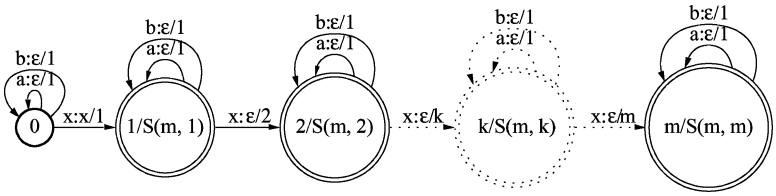

Figure 3. Weighted transducer T_m for computing the *m-th* moment of the count of $x \in \Omega$. The final weight at state k, $k = 1, \ldots, m$, is $S(m, k)$, the Stirling number of the second kind.

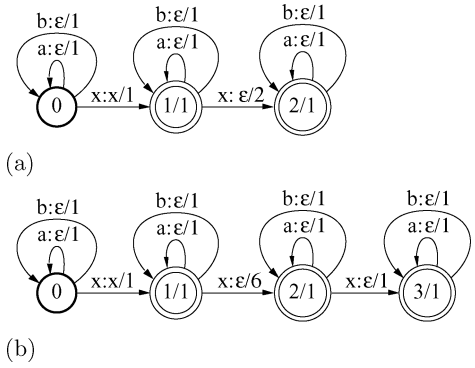

Figure 4. (a) Weighted transducer T_2 for computing the second moment. (b) Weighted transducer T_3 for computing the third moment.

input projection realizes α_m and its output is x. It admits $|\Sigma|$ self-loop transitions at each state. However, with a lazy implementation, these transitions can be explicitly constructed just when needed. The size of the lazy implementation of this transducer is in $O(m|x|)$, which proves the proposition. □

The transducer T_m of Figure 3 is quite simple and admits a natural lazy implementation. Figures 4(a) and (b) show the transducers T_2 and T_3 for computing the second and third moment of the counts of a sequence $x \in \Omega$.

4.2. General case

When the string x is not aperiodic, two consecutive occurrences of x may overlap. In that case, the counting transducer must be suitably modified to allow for such occurrences. Let U_k denote the set of all strings over the alphabet Σ with at least k occurrences of x, possibly overlapping. It is not hard to prove that U_k is a regular language.[6]

We can now generalize Proposition 1 to the case of all strings using arguments similar to those used in the aperiodic case.

Proposition 3. *Let $x \in \Sigma^*$. Then, for any positive integer m, there exists a weighted transducer T'_m such that for any weighted automaton A:*

$$[\![\Pi_2(A \circ T'_m)]\!](x) = c_m^A(x) \tag{19}$$

Proof: Let m be a positive integer. Let β_m be the weighted regular expression (or rational power series) defined by:

$$\beta_m = \sum_{k=0}^{m} k!\, S(m,k)\, U_k \tag{20}$$

The weight associated by β_m to the string $u \in \Sigma^*$ is:

$$(\beta_m, u) = \sum_{k=0}^{m} k!\, S(m,k)\, (U_k, u) \tag{21}$$

(U_k, u) represents exactly the number of ways of choosing k positions of x in u. The rest of the proof is similar to that of Proposition 1. □

Similarly, Proposition 2 can be generalized in the following way.

Proposition 4. *Let $x \in \Sigma^*$. There exists an algorithm for computing $c_m^A(x)$ for any weighted automaton A in $O(m|A||x|)$ time and $O(m|A||x|)$ space.*

Proof: By Proposition 3, $c_m^A(x)$ can be computed in $O(|A||T'_m|)$ time and space where T'_m is a weighted transducer with input automaton representing α_m and with output x. There exists such a weighted transducer with only $m|x| + 1$ states and admitting a lazy implementation whose size is in $O(m|x|)$. □

Figure 5 shows the weighted transducer T'_2 for the particular case of $x = aba$. x is not aperiodic. T'_2 only differs from T_2 by the transition from state 2 to state 4. This transition is used to account for *ababa* which contains two occurrences of *aba*.

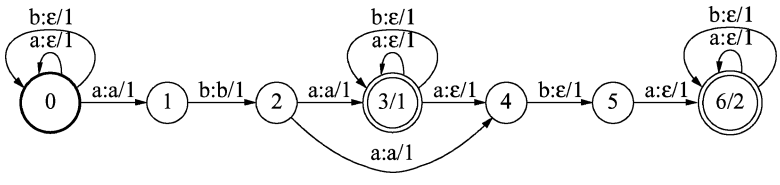

Figure 5. Weighted transducer T'_2 used to compute the second moment of the count of *aba* with the alphabet $\Sigma = \{a, b\}$.

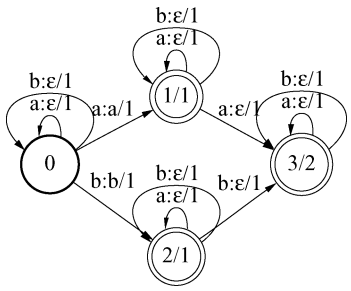

Figure 6. Weighted transducer T_2^X used to compute the second moment of the counts of all unigrams over the alphabet $\Sigma = \{a, b\}$.

4.2.1. Counts of a set of sequences. The computation of the kernels used in practice requires collecting not just the moments of the counts of a single sequence but those of a set of sequences $X \subseteq \Sigma^*$, e.g., the set of all n-gram sequences for a fixed n. When the set X is finite, a weighted transducer T_m^X for computing the counts of all sequences $x \in X$ can be constructed by simply taking the sum (union) of the counting transducers T_m defined for each $x \in X$. By definition of T_m^X, for all $x \in X$,

$$\left[\left[\Pi_2\left(A \circ T_m^X\right)\right]\right](x) = c_m^A(x) \tag{22}$$

Figure 6 shows the transducer T_2^X used to compute the second moment of the counts of all unigrams. This transducer has $|\Sigma| + 2$ states. But it admits a natural lazy implementation which avoids the explicit construction of the states and transitions corresponding to all elements of the alphabet when $|\Sigma|$ is large. More generally, the transducer T_m^X for computing the *m-th* moment of the counts of all unigram sequences has $(m-1)|\Sigma| + 2$ states but T_m^X admits a natural lazy implementation.

In the particular case of the expected counts, a simple transducer T_1^X can be constructed to compute the counts of all strings $x \in X$ for an arbitrary regular language X. Figure 7 shows the transducer T_1^X where the transition labeled with X serves as a shorthand for a finite automaton accepting X. It is clear that:

$$\left[\left[\Pi_2\left(A \circ T_1^X\right)\right]\right](x) = c_1(x) \tag{23}$$

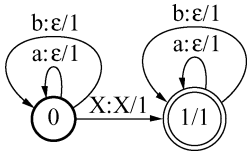

Figure 7. Weighted transducer T_1 used to compute the expected counts of all sequences $x \in L(X)$ with the alphabet $\Sigma = \{a, b\}$.

since for any $x \in L(X)$, $\Pi_2(T_1^X \circ x)$ coincides with the transducer T_1 defined for a single string x in Proposition 1. The size of a lazy implementation of the transducer T_1^X is in $O(A_X)$, where A_X is a finite automaton accepting X.

5. Application to spoken-dialog classification

We have fully implemented the core algorithms for computing the moment kernels described in the previous section using the AT&T FSM Library (Mohri, Pereira, & Riley, 2000) and the GRM Library (Allauzen, Mohri, & Roark, 2004). In particular, we implemented and used the *expectation kernels* $\mathcal{K}_1^n$ for various n-gram orders and the *variance kernel* $\mathcal{K}_2^1$ based on the second moment of the unigram sequences and applied them to several difficult spoken-dialog classification tasks.

5.1. Spoken-dialog classification problem

A key problem for the design of large-scale spoken-dialog systems is classification. This consists of assigning a specific category to each speech utterance. The categories help guide the dialog. There may be hundreds of categories depending on the task. They may correspond to the type of flight or billing mode in the case of a dialog with an airline reservation system, or to type of request, e.g., *refund*, or *calling-card* in the case of a dialog with an operator service system. Classification of a speech utterance is based on its transcription by a speech recognizer and is typically based on features such as some relevant word sequences.

Unfortunately, the word error rate of conversational speech recognition systems is still relatively high in many tasks. In the case of the deployed services we examined, it is about 30% when using the recognizer's most likely word transcription.[7] But, the full output of a speech recognizer contains more information. It is a *word lattice*, a weighted automaton compactly representing the recognizer's set of "best guesses", which contains the correct transcription in most cases. Each path of a word lattice is labeled with a sequence of words and has a weight that can be interpreted as a probability. The path with the highest probability is the recognizer's best guess. Thus, the objects to analyze for spoken-dialog classification are word lattices, i.e. distributions of word sequences given as weighted automata.

The design of classification algorithms for word lattices raises several issues. Word lattices, even relatively small ones, may contain more than a billion paths, thus classical algorithms devised for strings cannot be generalized by simply applying them to each path of the lattice. Furthermore, the paths are weighted and these weights must be used to guide appropriately the classification task. In previous work, we showed that the use of rational kernels solves both of these problems since they define kernels between weighted automata and since they can be computed efficiently.

The rational kernels previously introduced were based on the expected counts of sequences appearing in both automata. It is natural to assume indeed, as in the string case, that two lattices are similar when they share many common sequences. However, such kernels ignore higher order moments of the counts, which may be important to take into

consideration when comparing two word lattices. The moments kernels we defined in the previous sections generalize these kernels by taking into account higher order moments of the count distributions.

5.2. Experiments and results

We did a series of experiments with data collected from two deployed large-scale spoken-dialog systems using moments kernels. The next two sections describe our experiments and the corresponding results in detail.

5.2.1. HMIHY 0300. The first task we considered is that of a deployed customer-care application (HMIHY 0300). In this task, users interact with a spoken-dialog system via the telephone, speaking naturally, to ask about their bills, their calling plans, or other similar topics. Their responses to the open-ended prompts of the system are not constrained by the system, they may be any natural language sequence. The objective of the spoken-dialog classification is to assign one or several categories or call-types, e.g., *Billing Credit*, or *Calling Plans*, to the users' speech utterances.

We applied moment kernels to a difficult subset of a partition of the HMIHY 0300 task with 70 categories and a vocabulary size of 5,405 words for which the speech recognizer's word error rate is 28.7%. In our experiments, we used 10,794 word lattices as our training data and 2,784 lattices as our test data. Each utterance may be assigned to several classes and it is considered to be an error if the highest scoring class is not one of these labels.

We experimented with two types of moment kernels: expectation kernels, and kernels obtained by combining expectation kernels and variance kernels by summing their contributions, enhanced with polynomials of varying degrees d in the form $(1 + K)^d$. We applied SVMs with these kernels to the word lattices output by the speech recognition system and compared their results by varying the n-gram order and polynomial degree using the large-margin classification software library (LLAMA) written by Haffner, which includes an optimized multi-class logistic regression for recombination of the binary one-versus-rest SVMs. No attempt was made to optimize with respect to other SVM training parameters. Training and testing were done on a single processor of a 2.40 GHz Intel Pentium processor Linux cluster with 2 GB of memory and 512 KB cache. Training took in the order of 4–5 hours for both kernels.

The best results when using the expectation kernel alone were obtained with the n-gram order $n = 4$ and polynomial degree $d = 2$. In the case of variance kernels combined with expectation kernels, the best results were achieved with $n = 3$ and $d = 2$. The best result using variance kernels was clearly better than expectation kernels alone, which were previously shown to substantially improve on kernels based on the one-best path of the lattices. In particular, at 15% rejection rate, the error rate was reduced by 1% absolute, that is about 6.2% relative, which is significant in this task.[8] Figure 8 shows that this improvement is consistent in the range of 0–20% rejection and confirms the usefulness of higher-order moments of the counts of n-gram sequences.

Comparing the best result obtained with the expectation kernels alone versus the best one achieved with the combination of the expectation kernels and variance kernels may be subject to over-fitting on the test data. Thus, we did a second evaluation of the relative

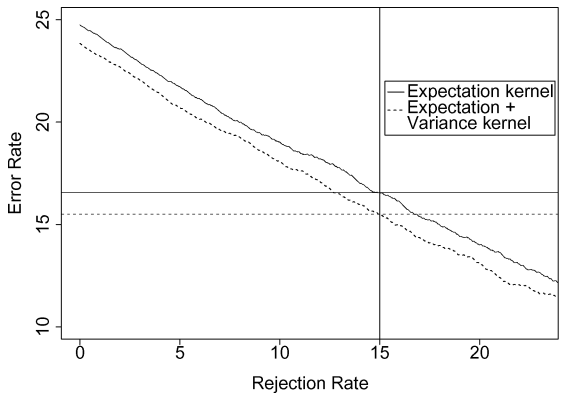

Figure 8. Experiments in 70-category HMIHY 0300 task. Comparison of the best result obtained using expectation kernels alone ($n = 4$, $d = 2$) applied to the one-best output of the recognizer and the best result achieved by combining expectation kernels with variance kernels ($n = 3$, $d = 2$).

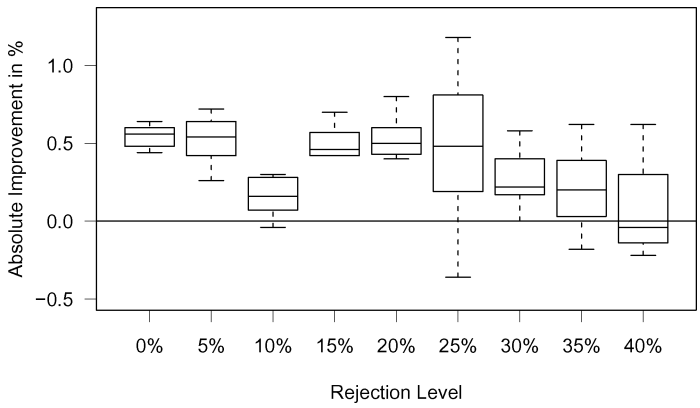

Figure 9. Classification improvement due to the use of the expectation kernel combined with the variance kernel over the use of the expectation kernel alone, in the 70-category HMIHY 0300 task. Performance improvement at various rejection levels for seven different parameter sets (n, d) are shown. For each set of parameters, training was done with the expectation kernel combined with the variance kernel and with the expectation kernel alone. The box-plot shows the average performance improvement (and its variance) when comparing the results with the same parameter settings at various rejection levels.

performance of these kernels by comparing their performance for the same settings of n and d, the set spanning performances from under-fitting ($n = 2$ and $d = 1$) to over-fitting ($n = 4$ and $d = 3$) on the training data. Our results show that the combined kernel systematically outperforms the expectation kernel (Figure 9). For each set of parameters, training was done separately with each kernel. The results were compared by plotting the performance improvement at various rejection levels. The box-plot is obtained by comparing results

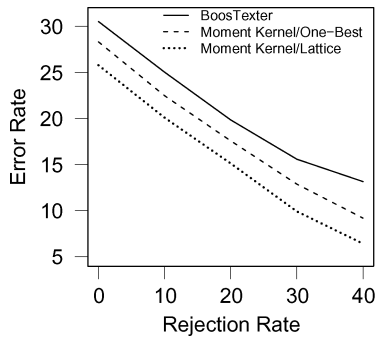

Figure 10. Experiments in a VoiceTone task. Comparison of the best previous classifier (BoosTexter) used in this task with the expectation kernel combined with the variance kernel ($n = 3$, $d = 2$) applied to the one-best output of the speech recognizer and to word lattices.

for seven different settings of (n, d). The consistent performance improvement up to 35% rejection level helps demonstrate the superior performance of the combined kernel.

5.2.2. VoiceTone2. Similarly, we did experiments with a more recently deployed spoken-dialog system (VoiceTone2) with a larger set of categories (82) and a higher word error rate (31.2%), both indicative of the greater difficulty of the classification task. In these experiments, we used about 9,000 word lattices as our training data and about 5,100 lattices as our test data. The average number of transitions of a word lattice in VoiceTone2 was about 360. These lattices may have more than a billion paths.

Our experiments show that expectation kernels combined with variance kernels lead to the best classification accuracy in this task with a performance that is substantially better than that of the best previous classifier designed for this task. The best previous classifier was the BoosTexter algorithm (Schapire & Singer, 2000) applied to the one-best hypothesis. Its results on the data sets we examined served as a baseline for our experiments. At about 15% rejection rate, the classification error is 5–6% (absolute value) lower than that of BoosTexter. Figure 10 presents the results of the experiments comparing these two classifiers. It also shows the accuracy achieved when applying these kernels to the one-best output of the speech recognizer. The substantial difference in accuracy between the plots (2–3% absolute value) demonstrates the benefit of the use of word lattices and that of kernels defined over such distributions.

Finally, as with HMIHY 0300, our experiments with VoiceTone2 showed that expectation kernels combined with variance kernels lead to an improvement of about 1% absolute value at 15% rejection rate with respect to the use of expectation kernels alone.

Altogether, our results show that the use of higher-order moment kernels consistently improves the classification accuracy in several difficult spoken-dialog tasks. The complexity of the computation of the moment kernels we used is no worse than that of the computation of expectation kernels and their implementation is quite similar. Thus, they can be of practical use for analyzing distributions of strings such as those found in natural language processing.

6. Conclusion

We introduced a new set of rational kernels that can be used to exploit higher-order moments of the counts of sequences appearing in weighted automata. We described efficient algorithms for computing these kernels using weighted finite-state transducers. We showed that they consistently improve the accuracy in several difficult spoken-dialog classification tasks using SVMs. Moment kernels can be applied similarly to many other text and speech processing tasks, and to other domains such as computational biology or web-log analysis.

Acknowledgments

We thank Patrick Haffner for his help with the use of the LLAMA software library.

Notes

1. Some kernels are also discussed by Gaertner, Flach, and Wrobel (2003) between "labeled directed graphs", i.e., finite automata. The kernels that these authors are striving to study are thus kernels between automata. Rational kernels provide a general solution for the definition of kernels between (weighted) automata and a general algorithm for their computation.
2. For a weighted automaton, weights are associated to paths and the expected count of a string is the weighted average of the number of occurrences of the string in a path.
3. See Pereira and Riley (1997); Mohri, Pereira, and Riley (1996) for a detailed presentation of the algorithm including the use of a transducer filter for dealing with the multiplicity of ϵ-paths.
4. Weighted transducers can be defined over arbitrary semirings. The weights are then multiplied along the paths using the second operation of the semiring and summed using the first operation. The composition algorithm described and the definition of rational kernels can also be extended to this more general setting (Cortes, Haffner, & Mohri, 2003a).
5. A string f is a factor of x if it a sequence of consecutive symbols appearing in x, that is if there exist x_1 and x_2 such that $x = x_1 f x_2$.
6. We can briefly sketch the construction of an automaton representing U_k. This also shows the regularity of U_k. Let A be the minimal deterministic automaton accepting $\Sigma^* x$, the set of strings over the alphabet Σ ending with x. We can construct from A a transducer T by augmenting all transitions of A with the output label ε and by inserting a transition with input ε and output #, a new symbol not in Σ, just before each transition leaving the unique final state of A, and by making all states final. When applied to any input string X, T outputs the symbol # and only that symbol exactly once after each occurrence of x. Let B be an automaton accepting $\#^k \#^*$. Then $\Pi_1(T \circ B)$ is an automaton accepting exactly the set of all strings with at least k occurrences of x.
7. The word error rate for one or several speech utterances is measured by the edit-distance between the word transcription given by the recognizer and the reference transcription for those utterances, divided by the total number of words in the reference transcription. Thus, roughly speaking, it measures the percentage of the words incorrectly transcribed by the recognizer.
8. Utterances whose score is lower than a given rejection threshold are rejected. With a rejection rate of $x\%$, the error rate is calculated by omitting the $x\%$ of the test examples with the lowest scores.

References

Allauzen, C., Mohri, M., & Roark, B. (2004). A general weighted grammar library. In *Proceedings of the Ninth International Conference on Implementation and Application of Automata (CIAA 2004)*. Kingston, Ontario, Canada. http://www.research.att.com/sw/tools/grm.

Boser, B. E., Guyon, I., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop of Computational Learning Theory*. (pp. 144–152). Vol. 5. Pittsburg ACM.

Cortes, C., Haffner, P., & Mohri, M. (2003a). Positive definite rational kernels. In *Proceedings of The 16th Annual Conference on Computational Learning Theory (COLT 2003)* (pp. 41–56). Vol. 2777 of *Lecture Notes in Computer Science*. Washington D.C., Germany: Springer, Heidelberg.

Cortes, C., Haffner, P., & Mohri, M. (2003b). Rational Kernels. In *Advances in Neural Information Processing Systems (NIPS 2002)* (Vol. 15). Vancouver, Canada: MIT Press.

Cortes, C., Haffner, P., & Mohri, M. (2003c). Weighted automata kernels—General framework and algorithms. In *Proceedings of the 9th European Conference on Speech Communication and Technology (Eurospeech '03), Special Session Advanced Machine Learning Algorithms for Speech and Language Processing*. Geneva, Switzerland.

Cortes, C. & Vapnik, V. N. (1995). Support-vector networks. *Machine Learning, 20:3*, 273–297.

Gaertner, T., Flach, P. A., & Wrobel, S. (2003). On graph kernels: Hardness results and efficient alternatives. In *Proceedings of The 16th Annual Conference on Computational Learning Theory (COLT 2003)*, Vol. 2777 of *Lecture Notes in Computer Science*. Washington D.C., Germany: Springer, Heidelberg.

Haussler, D. (1999). Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, University of California at Santa Cruz.

Leslie, C., Eskin, E., Weston, J., & Noble, W. S. (2003). Mismatch string kernels for SVM protein classification. In *NIPS 2002*. Vancouver, Canada: MIT Press.

Lodhi, H., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2001)., Text classification using string kernels. In T. K. Leen, T. G. Dietterich, & V. Tresp (Eds.), *NIPS 2000*. (pp. 563–569). MIT Press.

Mohri, M., Pereira, F. C. N., & Riley, M. (1996). Weighted automata in text and speech processing. In *ECAI-96 Workshop, Budapest, Hungary*: ECAI.

Mohri, M., Pereira, F. C. N., & Riley, M. (2000). The design principles of a weighted finite-state transducer library. *Theoretical Computer Science, 231*, 17–32.

Pereira, F. C. N. & Riley, M. D. (1997). Speech recognition by composition of weighted finite automata. In E. Roche & Y. Schabes (Eds.), *Finite-state language processing* (pp. 431–453). Cambridge, Massachusetts: MIT Press.

Schapire, R. E. & Singer, Y. (2000). BoosTexter: A boosting-based system for text categorization. *Machine Learning, 39:2/3*, 135–168.

Schölkopf, B. & Smola, A. (2002). Learning with kernels. Cambridge, MA: MIT Press.

Schützenberger, M. P. (1961). On the definition of a family of automata. *Information and Control, 4*.

Takimoto, E. & Warmuth, M. K. (2003). Path kernels and multiplicative updates. *The Journal of Machine Learning Research (JMLR), 4*, 773–818.

van Lint, J. H., & Wilson, R. M. (1992). *A course in combinatorics*. Cambridge University Press.

Vapnik, V. N. (1998). *Statistical learning theory*. John Wiley & Sons.

Watkins, C. (1999). Dynamic alignment kernels. Technical Report CSD-TR-98-11, Royal Holloway, University of London.

Received October 14, 2003
Revised May 19, 2004
Accepted May 24, 2004