Distribution Kernels Based on Moments of Counts

Corinna Cortes CORINNA@GOOGLE.COM
Google Labs, 1440 Broadway, New York, NY 10018

Mehryar Mohri MOHRI@RESEARCH.ATT.COM
AT&T Labs – Research, Shannon Laboratory
180 Park Avenue, Florham Park, NJ 07932-0971

Abstract

Many applications in text and speech processing require the analysis of distributions of variable-length sequences. We recently introduced a general kernel framework, *rational kernels*, to extend kernel methods to the analysis of such variable-length sequences or more generally weighted automata. These kernels are efficient to compute and have been successfully used in applications such as spoken-dialog classification using Support Vector Machines.

However, the rational kernels previously introduced do not fully encompass distributions over alternate sequences. Prior similarity measures between two weighted automata are based only on the expected counts of co-occurring subsequences and ignore similarities (or dissimilarities) in higher order moments of the distributions of these counts.

In this paper, we introduce a new family of rational kernels, *moment kernels*, that precisely exploit 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.

Appearing in *Proceedings of the 21st International Conference on Machine Learning*, Banff, Canada, 2004. Copyright 2004 by the owner.

1. Introduction

Many applications in text and speech processing require the analysis of distributions of variable-length sequences. The output of complex systems combining multiple information sources, e.g., large-vocabulary speech recognition or information extraction systems, 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.

An approach widely used in statistical learning techniques is that of kernel methods due to their computational efficiency in high-dimensional feature spaces (Schölkopf & Smola, 2002). Recently, a general kernel framework based on weighted transducers, *rational kernels*, was introduced to extend kernel methods to distributions represented by weighted automata (Cortes et al., 2003b; Cortes et al., 2003a). The framework was shown to include many string kernels introduced for text classification or computational biology. It was shown that there are general and efficient algorithms for computing rational kernels. Rational kernels have been combined with Support Vector Machines (SVMs) (Boser et al., 1992; Cortes & Vapnik, 1995; Vapnik, 1998) and successfully used in many applications including spoken-dialog classification and computational biology.

However, the rational kernels previously introduced do not fully encompass distributions over alternate sequences. Prior similarity measures between two weighted automata are based only on the expected counts of co-occurring subsequences and ignore similarities (or dissimilarities) in higher order moments of the distributions of these counts.[1] In this paper, we in-

[1]For a weighted automaton, the expected count of a string is the weighted average of the number of times the string appears in a path.

troduce a new family of rational kernels, *moment kernels*, that precisely exploit 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.

This paper describes in detail the algorithms for computing the moments of the count of an arbitrary string in a weighted automaton and gives the proof of their correctness. Rational kernels exploiting moments of the counts of all n-gram sequences are then defined. These moment kernels generalize those based on just the expected counts. The last section details their application to spoken-dialog classification and reports the results of our experiments in several difficult tasks.

2. Preliminaries

This section presents the definitions and notation necessary to present the algorithms and kernels described in the next sections. We give a brief introduction to weighted automata and weighted finite-state transducers. We refer the reader to (Mohri, 1997) for an extensive presentation of these devices.

A *weighted finite-state transducer* T is an 8-tuple $T = (\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$ where Σ is the finite input alphabet of the transducer, Δ is the finite output alphabet, Q is a finite set of states, $I \subseteq Q$ the set of initial states, $F \subseteq Q$ the set of final states, $E \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Delta \cup \{\epsilon\}) \times \mathbb{R} \times Q$ a finite set of transitions 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 a 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 a pair of strings $(x, y) \in \Sigma^* \times \Delta^*$:

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

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 a string $x \in \Sigma^*$:

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

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 $\Pi_2(T)$ the weighted automaton obtained from T by removing its input labels.

A general *composition* operation similar to the composition of relations can be defined for weighted finite-state transducers. The composition of two transducers T_1 and T_2 is a weighted transducer denoted by $T_1 \circ T_2$ and defined by:

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

There exists an algorithm for computing and constructing $T = T_1 \circ T_2$ from T_1 and T_2 (Pereira & Riley, 1997; Mohri et al., 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 and a transition of T_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. Moments of Counts of Strings

This section gives the definition of the moments of the distribution of the counts of an arbitrary sequence x appearing in a weighted automaton A, describes efficient algorithms for computing it, 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.

3.1. Definition

Let $A = (Q, I, F, \Sigma, \delta, \sigma, \lambda, \rho)$ 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.

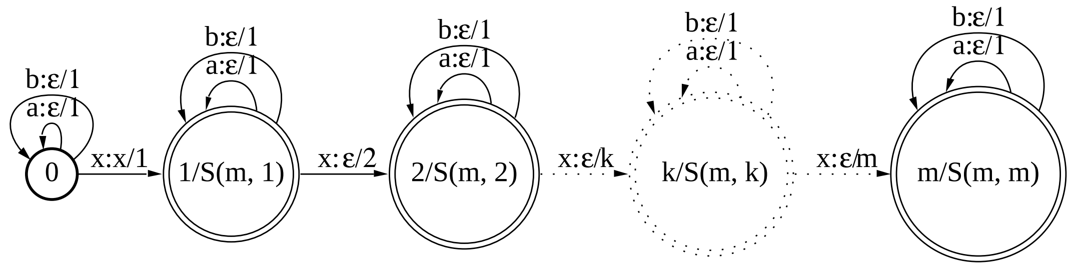

Figure 1. Weighted transducer T_m for computing the m-th moment of the count of $x \in \Omega$. An initial state is represented by bold circles, final states by double circles. Inside each circle, the first number indicates the state number, the second, at final states only, the value of the final weight function ρ at that state. The final weight at state k, $k = 1, \ldots, m$, is $S(m, k)$, the Stirling numbers of the second kind. Each transition is labeled with an input label and an output label separated by a colon, and carries some weight indicated after the slash symbol, e.g., the transition from state 1 to state 2 has input label x, output label ϵ (the empty string), and weight 2.

Definition 1 *Let $m \geq 1$. The* m-th moment of the count *of the sequence x in A, is denoted by $c_m(x)$ and defined as:*

$$c_m(x) = \sum_{u \in \Sigma^*} |u|_x^m \, [\![A]\!](u) \tag{4}$$

where $|u|_x$ denotes the number of occurrences of x in the string u labeling one or several successful paths of A.

We will 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.

3.2. Algorithms

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 string over the alphabet Σ.

3.2.1. 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:*

$$[\![\Pi_2(A \circ T_m)]\!](x) = c_m(x) \tag{5}$$

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

$$\alpha_m = \sum_{k=0}^{m} k! \, S(m,k) \, (\Sigma^* x)^k \Sigma^* \tag{6}$$

where $S(m,k)$, $k = 1, \ldots, m$, denote the Stirling numbers of the second kind (van Lint & Wilson, 1992). 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:

$$(\alpha_m, u) = \sum_{k=0}^{m} k! \, S(m,k) \, ((\Sigma^* x)^k \Sigma^*, u) \tag{7}$$

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

$$(\alpha_m, u) = \sum_{k=0}^{m} \binom{N}{k} k! \, S(m,k) = N^m \tag{8}$$

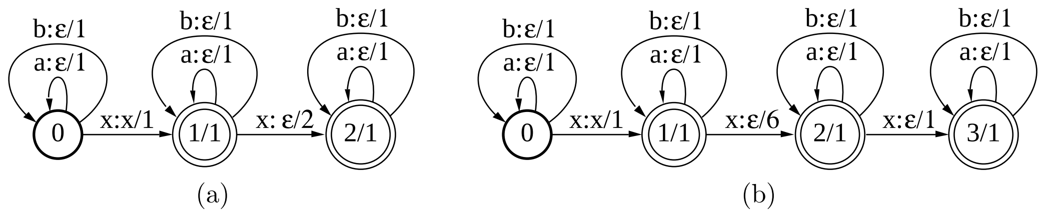

Figure 2. (a) Weighted transducer T_2 for computing the second moment. (b) Weighted transducer T_3 for computing the third moment.

where we use a standard formula relating Stirling numbers of the second kind (van Lint & Wilson, 1992).

Since α_m is a weighted regular expression, by the theorem of (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 exactly verifies 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 $c_m(x)$ for any weighted automaton A in $O(m|A||x|)$ time and $O(m|A||x|)$ space.*

Proof. By Proposition 1, $c_m(x)$ 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. Figure 1 shows that transducer. Clearly, its 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 1 is quite simple and admits a natural lazy implementation. Figures 2(a)-(b) show the transducers T_2 and T_3 for computing the second and third moment of the counts of a sequence $x \in \Omega$.

3.2.2. General Case

When the string x is not aperiodic, two consecutive occurrences of x may have an 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. 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:*

$$[\![\Pi_2(A \circ T'_m)]\!](x) = c_m(x) \tag{9}$$

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

$$\beta_m = \sum_{k=0}^{m} k!\, S(m,k)\, U_k \tag{10}$$

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

$$(\beta_m, u) = \sum_{k=0}^{m} k!\, S(m,k)\, (U_k, u) \tag{11}$$

(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 $c_m(x)$ for any weighted automaton A in $O(m|A||x|)$ time and $O(m|A||x|)$ space.*

Proof. By Proposition 3, $c_m(x)$ 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 3 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.

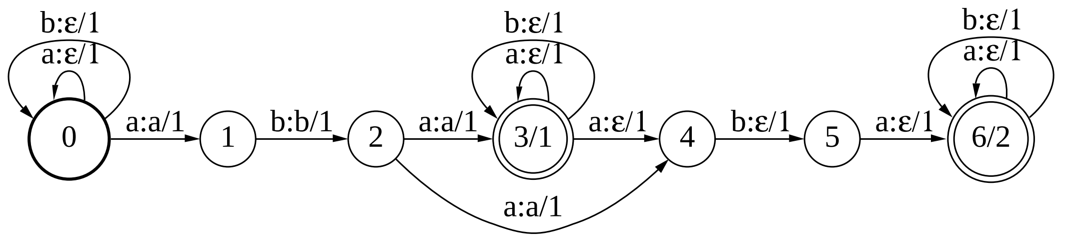

Figure 3. Weighted transducer T'_2 used to compute the second moment of the count of *aba* with the alphabet $\Sigma = \{a, b\}$.

3.2.3. 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 $x \in X$,

$$[\![\Pi_2(A \circ T_m^X)]\!](x) = c_m(x) \tag{12}$$

Figure 4 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 n-gram sequences has $(|\Sigma| + 1)^m + 1$ 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 5 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:

$$[\![\Pi_2(A \circ T_1^X)]\!](x) = c_1(x) \tag{13}$$

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.

4. Moment Kernels

The algorithms given in the previous section can be used to compute rational kernels exploiting other moments of the distributions given as weighted automata.

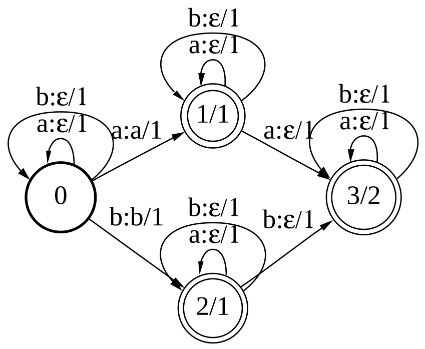

Figure 4. Weighted transducer T_2^X used to compute the second moment of the counts of all unigrams over the alphabet $\Sigma = \{a, b\}$.

In previous work (Cortes et al., 2003b; Cortes et al., 2003c), we defined n-gram kernels K_1^n based on the expected counts of common n-gram sequences between two automata A_1 and A_2:

$$K_1^n(A_1, A_2) = \sum_{|x|=n} c_1^{A_1}(x) c_1^{A_2}(x) \tag{14}$$

where $c^{A_i}(x)$, $i = 1, 2$, 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.

We now introduce a general family of kernels, *moment kernels*, denoted by K_m^n, $m, n \geq 1$ and defined by:

$$K_m^n(A_1, A_2) = \sum_{|x|=n} c_m^{A_1}(x) c_m^{A_2}(x) \tag{15}$$

By Propositions 1-4, moment kernels are rational kernels and can be computed efficiently using the algorithms previously described. Indeed,

$$K_m^n(A_1, A_2) = w[A_1 \circ (T_m^n \circ T_m^{n\,-1}) \circ A_2] \tag{16}$$

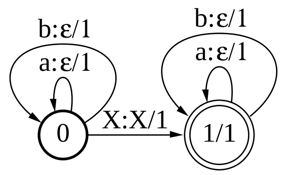

Figure 5. Weighted transducer T_1 used to compute the expected counts of all sequences $x \in L(X)$ with the alphabet $\Sigma = \{a, b\}$.

where T_m^n is the weighted transducer counting the m-th moment of the counts of all n-gram sequences over the alphabet Σ. Since their definition is based on weighted transducers of the type $T \circ T^{-1}$, moment kernels are positive definite (Cortes et al., 2003a). This guarantees the convergence of training for discriminant classification algorithms such as SVMs when using these kernels.

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 et al., 2003a), these 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. The next section reports the results of our preliminary experiments in several difficult spoken-dialog classification tasks using these distribution kernels.

5. Spoken-Dialog Application – Experimental Results

We have fully implemented the core algorithms for computing the moment kernels described in the previous section using the AT&T FSM Library (Mohri et al., 2000) and the GRM Library (Allauzen et al., 2003). In particular, we implemented and used the *expectation kernels* K_n^1 for various n-gram orders and the *variance kernel* K_1^2 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, out of a set of about hundred classes, a specific category to each speech utterance. Categories help guide the dialog. Their definition depends 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 relevant word sequences.

The word error rate of conversational speech recognition systems is still relatively high in many tasks. In the case of the deployed services we experimented with, it is about 30% if one relies only on the one-best output of the recognizer. But, the full output of a speech recognizer is a *word lattice*, a weighted automaton compactly representing the recognizer's "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 largest 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 to compare two word lattices. The moments kernels we defined in 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, 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 P. Haffner which includes an optimized multi-class recombination of binary 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 2.40GHz Intel Pentium processor Linux cluster with 2GB 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 sequences of 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. Figure 6 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.

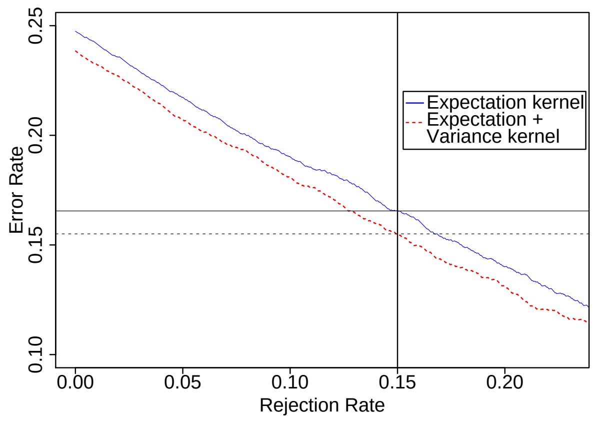

Figure 6. Experiments in 70-category HMIHY 0300 task. Comparison of the best results obtained using expectation kernels alone ($n = 4$, $d = 2$) applied to the one-best output of the recognizer and the best result achieved by combining expectation kernels with variance kernels ($n = 3$, $d = 2$).

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.

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. At about 15% rejection rate, the classification error is 5-6% lower than that of the best previous classifier. Figure 7 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-

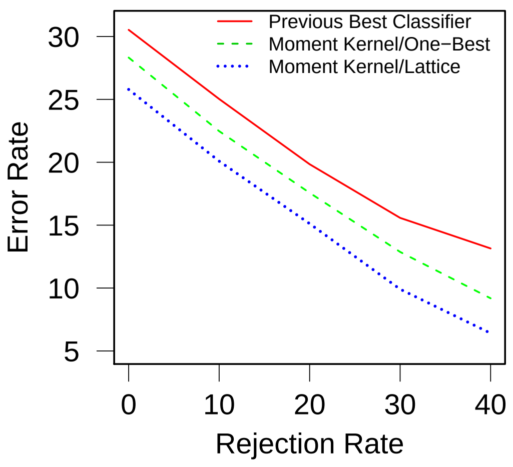

Figure 7. Experiments in a VoiceTone task. Comparison of the best previous classifier used in this task with the expectation kernel combined with the variance kernel ($n = 3$, $d = 2$) applied to the one-best output of the speech recognizer and to word lattices.

order moment kernels consistently improves the classification accuracy in several difficult spoken-dialog tasks. The complexity of the computation of these moment kernels 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 task using SVMs. Moment kernels can be applied similarly to many other text and speech processing tasks, and to other domains such as computational biology.

7. Acknowledgments

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

References

Allauzen, C., Mohri, M., & Roark, B. (2003). General-Purpose Grammar Library – GRM Library. *http://www.research.att.com/sw/tools/grm, AT&T Labs - Research.*

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

Cortes, C., Haffner, P., & Mohri, M. (2003a). Positive Definite Rational Kernels. *Proceedings of The 16th Annual Conference on Computational Learning Theory (COLT 2003)* (pp. 41–56). Washington D.C.: Springer, Heidelberg, Germany.

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

Cortes, C., Haffner, P., & Mohri, M. (2003c). Weighted Automata Kernels – General Framework and Algorithms. *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*, 273–297.

Mohri, M. (1997). Finite-state transducers in language and speech processing. *Computational Linguistics*, *23:2*.

Mohri, M., Pereira, F. C. N., & Riley, M. (1996). Weighted automata in text and speech processing. *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. http://www.research.att.com/sw/tools/fsm.

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

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

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

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.