Weighted Automata Kernels – General Framework and Algorithms

Corinna Cortes, Patrick Haffner, Mehryar Mohri

AT&T Labs – Research
180 Park Avenue, Florham Park, NJ 07932, USA
{corinna, haffner, mohri}@research.att.com

Abstract

Kernel methods have found in recent years wide use in statistical learning techniques due to their good performance and their computational efficiency in high-dimensional feature space. However, text or speech data cannot always be represented by the fixed-length vectors that the traditional kernels handle. We recently introduced a general kernel framework based on weighted transducers, *rational kernels*, to extend kernel methods to the analysis of variable-length sequences and weighted automata [5] and described their application to spoken-dialog applications. We presented a constructive algorithm for ensuring that rational kernels are *positive definite symmetric*, a property which guarantees the convergence of discriminant classification algorithms such as Support Vector Machines, and showed that many string kernels previously introduced in the computational biology literature are special instances of such positive definite symmetric rational kernels [4]. This paper reviews the essential results given in [5, 3, 4] and presents them in the form of a short tutorial.

1. Introduction

Traditionally, statistical learning techniques have focused on classification of fixed-length input vectors describing features of the objects to be classified. However, in many applications such as speech recognition and computational biology, the objects to study and classify are not just fixed-length vectors, but variable-length sequences. Sometimes they are even large sets of alternative sequences with associated probabilities. Consider for example the problem of classifying speech recognition outputs in a large spoken-dialog application. For a given speech utterance, the output of a large-vocabulary speech recognition system is a weighted automaton called a *word lattice* compactly representing the possible sentences and their respective probabilities based on the models used. Such lattices, while containing sometimes just a few thousand transitions, may contain hundreds of millions of paths each labeled with a distinct sentence and their associated probabilities.

The application of discriminant classification algorithms to word lattices, or more generally weighted automata, raises two issues: that of handling variable-length sequences, and that of applying a classifier to a distribution of alternative sequences. A common approach to the variable-length problem is to extract fixed-length feature vectors from the lattices. Examples are frequency counts for a fixed bag of words or frequency counts for n-grams. However, computing such features easily becomes infeasible, and yet they only approximate the full richness of the lattices. The most common approach to the alternative sequences is the Viterbi approximation that simply keeps the highest scoring path and disregards the others.

The work we describe here is a technique that is very general and addresses both of these problems. Our technique operates on the full lattice without the need for extracting feature vectors or disregarding alternative sequences. It is computationally efficient and it is based on well-known general weighted automata algorithms that have been extensively used in speech and language applications. The work was originally reported in [5, 3, 4]. This paper presents essential parts of our work in the form of a short tutorial.

Our approach is based on kernels. Kernels are in general used for computing inner products or similarity measures between feature vectors in a closed form without explicitly extracting the feature vectors. Kernel methods are widely used in statistical learning techniques such as Support Vector Machines (SVMs) [21] due to their computational efficiency in high-dimensional feature spaces. This motivates the introduction and study of kernels for weighted automata. We present a general family of kernels based on weighted transducers or rational relations, *rational kernels* [5] which apply to weighted automata.

Section 3 introduces rational kernels for weighted automata, and shows that they can be computed efficiently using a general algorithm of composition of weighted transducers and a general single-source shortest-distance algorithm. Then, it shows how to build rational kernels that are positive definite and symmetric, properties needed to ensure proper convergence when combined with machine learning techniques such as SVMs.

Section 4 demonstrates some specific rational kernels and their applications to spoken-dialog classification, where significant improvements are observed.

2. Kernel Methods

In most classification tasks, input vectors $x \in X$ are not linearly separable, but a non-linear mapping $\Phi : X \to F$ can be defined from the original set to a typically higher-dimensional feature set F such that the images $\Phi(x)$, $x \in X$, are linearly separable. This high-dimensional feature space can for instance consist of all pairwise correlations between elements of x.

Training and classification with classification techniques such as SVMs require the computation of the inner product between two vectors, which in the case of a high-dimensional space such as F may very costly. A solution to this problem is to use instead a *kernel*, that is a function $K : X \times X \to \mathbb{R}$, such that for all $x, y \in X$:

$$K(x, y) = (\Phi(x), \Phi(y)) \tag{1}$$

Kernels may be more efficient to compute, in practice often substantially more efficient, because they do not require the explicit computation of Φ. There is some flexibility in the choice of the kernel function K, provided that there exists a mapping Φ such that Equation 1 holds. Functions K that verify this condition must be *positive definite symmetric* (PDS) [1]. This condition guarantees the convergence to a global optimum of discriminant classification algorithms such as SVMs. PDS kernels K can be used to construct other families of PDS kernels, e.g., *polynomial kernels* of degree p defined by $(K + a)^p$, or *Gaussian kernels* defined by $\exp(-d^2/\sigma^2)$ with $d^2(x, y) = K(x, x) + K(y, y) - 2K(x, y)$ [20].

Classification algorithms such as SVMs have been primarily applied to input vectors $x \in X$ of fixed length or structure. In the following, we present a general kernel framework for variable-length sequences, or, more generally, for weighted automata.

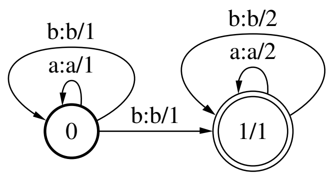

Figure 1: Weighted transducer computing the integer value of binary numbers ($a = 0$ and $b = 1$) [6].

3. Rational Kernels

Let us first introduce a general and computationally efficient framework for kernels between strings over a given alphabet Σ.

Kernels between strings can define arbitrarily complex functions. The weighted relation that a kernel computes may correspond to an algebraic power series [11], i.e. have the complexity of a context-free language, or it may correspond to even more powerful or complex languages. The time complexity of the computation of such powerful kernels may be cubic or more in the size of each string, which, in general, makes them unsuitable for many real-world applications.

This brings us to consider kernels that are more efficient to compute and that yet have a wide coverage. We define a family of kernels, *rational kernels*, whose definition is based on *rational transductions*, or *weighted finite-state transducers*.

3.1. Weighted Finite-State Transducers

Figure 1 shows an example of a weighted finite-state transducer. A weighted transducer is a weighted directed graph whose edges, called *transitions*, are labeled with both an input, an output symbol, and a weight. Input and output labels are in the figure separated by a colon, and the weight is indicated after the slash separator. A weighted transducer has a set of initial states represented in the figure by a bold circle, and a set of final states, represented by double circles. A path from an initial state to a final state is a *successful path*.

The weight of a path is obtained by multiplying the weights of its constituent transitions. The weight of a successful path is multiplied by the weight of the initial state (typically equal to one in all our examples) of the path and the weight of the final state of the path (marked after slash). The weight associated by a weighted transducer T to a pair of strings $(x, y) \in \Sigma^* \times \Sigma^*$ is denoted by $[\![T]\!](x, y)$ and is obtained by taking the sum of the weights of all successful paths with input label x and output label y.[1] For example, the transducer of Figure 1 associates the weight 3 to the pair (bb, bb) since there are two successful paths labeled with bb: one with weight 1 and another one with weight 2. More generally, the transducer of Figure 1 associates to each pair (x, x) with $x \in \{a, b\}^*$, the integer value of x viewed as a binary number with $a = 0$ and $b = 1$ [6].

3.2. Algorithms

There exists a simple and efficient algorithm, *composition*, for computing $[\![T]\!](x, y)$. The composition of two weighted transducers T_1 and T_2 is a weighted transducer denoted by $T_1 \circ T_2$ whose paths π are obtained by matching the output labels of a path π_1 of T_1 with the input labels of path π_2 of T_2, with the weight of π obtained by taking the product of the weights of the paths π_1 and π_2 [2, 7, 18, 11]. Thus, the weight associated by $T_1 \circ T_2$ to the pair (x, y) is:

$$[\![T_1 \circ T_2]\!](x, y) = \sum_{z \in \Sigma^*} [\![T_1]\!](x, z) \times [\![T_2]\!](z, y) \qquad (2)$$

The algorithm for constructing $T_1 \circ T_2$ is based on matching the transitions of T_1 and T_2. States in the composition $T_1 \circ T_2$ can be identified with pairs of a state of T_1 and a state of T_2.

[1]These definitions and the following algorithms can be generalized to the case where the weights are element of an arbitrary *semiring*.

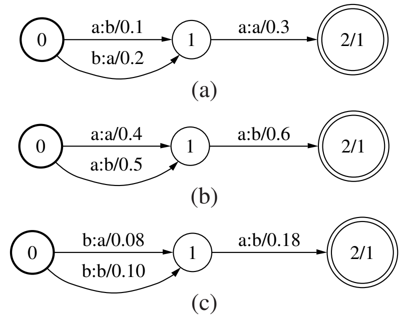

Figure 2: *Composition of weighted transducers. The transducer of Figure (c) is obtained by composing those of Figures (a)-(b).*

Let (q_1, a, b, w_1, q_2) be a transition of T_1 from state q_1 to q_2 with input label a, output label b, and weight w_1, and similarly (q'_1, b, c, w_2, q'_2) a transition of T_2 from state q'_1 to q'_2, then the algorithm constructs the following transition from state (q_1, q'_1) to state (q_2, q'_2):

$$((q_1, q'_1), a, c, w_1 \times w_2, (q_2, q'_2)) \qquad (3)$$

The case of transitions with ϵ inputs or outputs requires additionally the use of a transducer filter to deal with the multiplicity of ϵ-paths [17, 15]. Figures 2 (a)-(c) illustrate the algorithm when applied to the transducers of Figures 2(a)-(b). In the worst case, all transitions of T_1 leaving a state q_1 match all those of T_2 leaving state q'_1, thus the space and time complexity of composition is quadratic: $O(|T_1||T_2|)$, where $|T_i|$ is the sum of the number of states and transitions of the transducer T_i.

A string x can be represented by a linear finite automaton A_x accepting it. To determine the paths of T with input label x and output label y, we can compute the composition machine:

$$M = A_x \circ T \circ A_y \qquad (4)$$

and then compute $w[M]$, the sum of the weights of all paths of M by using the forward-backward algorithm when M is acyclic, or a general *shortest-distance algorithm* that works with many semirings [14].

3.3. Definition of Rational Kernels

We can use the composition operation to define a family of kernels for weighted automata. We denote by Ω the set of weighted automata over the alphabet Σ.

Definition 1 *$K : \Omega \times \Omega \to \mathbb{R}$ is a* rational kernel *if there exists a weighted transducer T and a function $\psi : \mathbb{R} \to \mathbb{R}$ such that for all $X, Y \in \Omega$:*

$$K(X, Y) = \psi(w[X \circ T \circ Y]) \qquad (5)$$

Note that we are not making any particular assumption about the function ψ in this definition. In most cases of interest, however, ψ is simply the identity function. Note that a rational kernel K also defines a kernel between strings. That corresponds to the specific case where the automata X and Y represent each just a single string. Since the worst cost complexity of composition is quadratic, assuming that ψ can be computed in constant time, the cost of the computation of $K(X, Y)$ is in $O(|T||X||Y|)$. This complexity can be substantially improved in some cases of interest as we will see in Section 5.

An example of rational kernel is the gappy n-gram kernel with decay factor λ defined by [13]. Figure 3 shows the corresponding weighted transducer for $n = 2$ and $\Sigma = \{a, b\}$. Gappy kernels are PDS kernels and thus verify the condition mentioned in Section 2 [5].

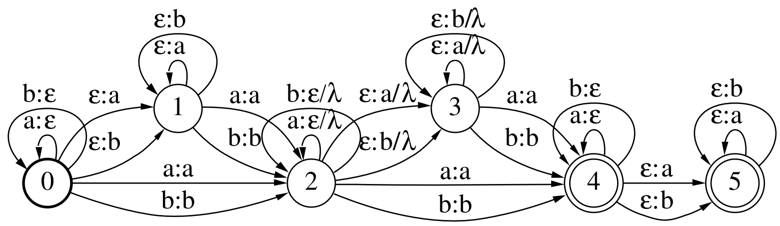

Figure 3: Gappy bigram kernel with decay factor λ. All unmarked transition weights are equal to 1.

3.4. PDS Rational Kernels

Rational kernels define a general set of kernels based on weighted transducers that can be computed efficiently using composition. But, not all rational kernels verify the conditions mentioned in Section 2. How can we design a PDS rational kernel? Assume that the function ψ in the definition of rational kernels is a continuous morphism. Then, the following result [4] shows that one can in fact construct a PDS rational kernel from an arbitrary weighted transducer T. The construction is based on the composition of T with its inverse T^{-1}, that is the transducer obtained from T by swapping input and output labels of each transition.

Theorem 1 *Let T be a weighted transducer and assume that the weighted transducer $T \circ T^{-1}$ is well-defined, then $T \circ T^{-1}$ defines a PDS rational kernel over $\Sigma^* \times \Sigma^*$.*

Many string kernels introduced in the computational biology literature are special instances of PDS rational kernels of the type $T \circ T^{-1}$ [22, 10, 12].

Complex PDS rational kernels can also be constructed from simpler ones. Indeed, weighted transducers are closed under sum, product, and closure [2]. When the same function ψ is used in the definition of all rational kernels and ψ is a continuous morphism, PDS rational kernels are also closed under those operations [4]. In particular, let T_1 and T_2 be the associated transducers of two PDS rational kernels K_1 and K_2 defined with the same morphism ψ, then:

$$(K_1 + K_2)(x, y) = \psi([\![(T_1 \oplus T_2)]\!](x, y)) \tag{6}$$

and $(K_1 + K_2)$ is a PDS rational kernel.

4. Spoken-Dialog Applications

This section introduces some PDS rational kernels relevant to spoken-dialog classification tasks and that are used in our experiments.

4.1. Definition of the Kernels

A rational kernel can be viewed as a similarity measure between two sequences or weighted automata. One may for example consider two utterances to be similar when they share many common n-gram subsequences. The exact transcriptions of the utterances are not available but we can use the word lattices output by the recognizer instead which are weighted automata called word lattices.

A word lattice L can be viewed as a probability distribution P_L over all strings $s \in \Sigma^*$. Denote by $|s|_x$ the number of occurrences of a sequence x in the string s. The expected count or number of occurrences of an n-gram sequence x in s for the probability distribution P_L is:

$$c(L, x) = \sum_s P_L(s)|s|_x$$

Two lattices output by a speech recognizer can be viewed as similar when the sum of the product of the expected counts they assign to their common n-gram sequences is sufficiently high.

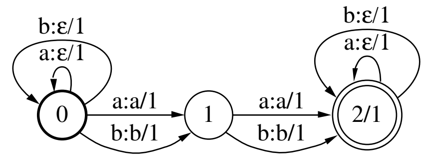

Figure 4: Weighted transducer T computing expected counts of bigram sequences of a word lattice with $\Sigma = \{a, b\}$.

Thus, we define an n-gram kernel K for two lattices L_1 and L_2 by:

$$K(L_1, L_2) = \sum_{|x|=n} c(L_1, x)\, c(L_2, x) \tag{7}$$

It is already clear from this expression that K is a PDS kernel since it is written as a dot product. We can show that K is a rational kernel and that it can be computed efficiently.

Indeed, there exists a simple weighted transducer T that can be used to computed $c(L_1, x)$ for all n-gram sequences $x \in \Sigma^*$ [3]. Figure 4 shows that transducer in the case of bigram sequences ($n = 2$) and for the alphabet $\Sigma = \{a, b\}$. The general definition of T is:

$$T = (\Sigma \times \{\epsilon\})^* \, (\sum_{a \in \Sigma} \{a\} \times \{a\})^n \, (\Sigma \times \{\epsilon\})^* \tag{8}$$

The kernel K can be written in terms of the weighted transducer T as [3]:

$$\begin{aligned} K(L_1, L_2) &= w[(L_1 \circ T) \circ (T^{-1} \circ L_2)] & (9) \\ &= w[(L_1 \circ (T \circ T^{-1}) \circ L_2)] & (10) \end{aligned}$$

which shows that it is a rational kernel whose associated weighted transducer is $T \circ T^{-1}$.

4.2. Computation

The general composition algorithm and shortest-distance algorithm described in Section 3.2 can be used to compute K efficiently. The size of the transducer T is $O(n|\Sigma|)$ but in practice, a lazy implementation can be used to simulate the presence of the transitions of T labeled with all elements of Σ. This reduces the size of the machine used to $O(n)$. Thus, since the complexity of composition is quadratic [15, 17] and since the general shortest distance algorithm just mentioned is linear for acyclic graphs such as the lattices output by speech recognizers [14], the worst case complexity of the algorithm is: $O(n^2 \, |L_1| \, |L_2|)$. In the case where the lattices L_1 and L_2 represent each just a unique string, a more efficient algorithm based on the notion of *failure function* can be used and the total complexity of the computation is then: $O(n + |L_1| + |L_2|)$ [3].

5. Experimental Results

We used the AT&T FSM Library [16] for the implementation of our lattice kernel and incorporated our algorithm in a general learning library for large-margin classification, LLAMA, [9]. LLAMA offers an optimal multi-class recombination of binary SVMs where kernels are optimized separately for each class.

We used the lattice kernel defined in the previous section for call-classification in the spoken language understanding (SLU)

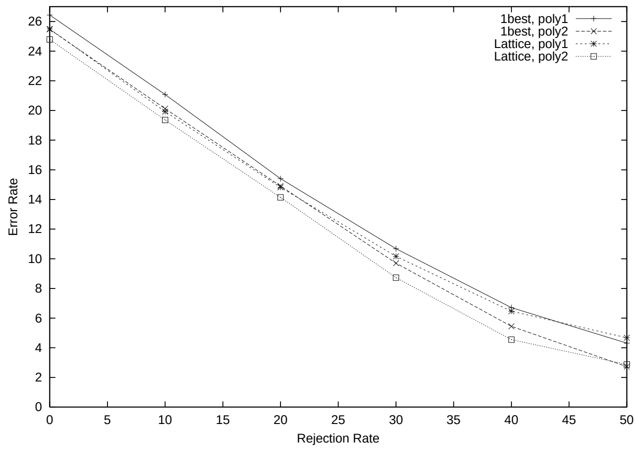

Figure 5: Comparison of SVMs trained with trigram kernel applied to the full lattice versus the one-best path.

component of the AT&T *How May I Help You* (*HMIHY*^{SM}) natural dialog system. In this system, users ask questions about their bill, calling plans, etc. The utterances are the responses to the first greeting: "*Hello, This is AT&T. How May I Help You?*". The word accuracy for these utterances with the current recognition system is 72.5%.

Within the SLU component, the objective is to classify the input telephone call into one of 70 classes (call-types and named entities), such as *Billing Credit*, or *Calling Plans* [8]. 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. In our experiments, we used 10,794 utterances as our training data and 2,784 utterances as our test data. The feature space corresponding to our lattice kernel is that of all possible trigrams over a vocabulary of 5,405 words. Training required just a few minutes on a single processor of a 1GHz Intel Pentium processor Linux cluster with 2GB of memory and 256 KB cache.

Utterances that score lower than a given confidence level are rejected. We present our results in the form of a plot of the classification error as a function of the rejection rate. Figure 5 shows the comparison of classification based on the full lattice versus the one-best path. The lattice kernel brings a significant reduction of the one-error rate from 9.7% to 8.7% at a rejection level of 30% ('1best, poly2' versus 'Lattice, poly2'), a typical operation point for the task. The plot also demonstrates how a 2-degree polynomial applied to the lattice kernel further reduces the classification error ('Lattice, poly1' versus 'Lattice, poly2').

6. Conclusion

We have reviewed essential results for rational kernels, introduced to address the study and classification of variable length sequences or even distributions over sets of alternative sequences. These kernels can be computed using general weighted automata and graph algorithms, which avoid the design of *ad hoc* or special-purpose algorithms. They can be defined and combined using general weighted automata algorithms. These algorithm are very well studied and often highly efficient. They are implemented in the AT&T FSM Library [16] and are available for download for non-commercial use.

The family of PDS rational kernels includes many of the kernels used in computational biology or text processing applications, and we demonstrated how significant performance improvement can be obtained using one of these kernels on a multi-class call-classification problem.

7. References

[1] C. Berg, J. P. R. Christensen, and P. Ressel. *Harmonic Analysis on Semigroups*. Springer-Verlag: Berlin-New York, 1984.

[2] J. Berstel. *Transductions and Context-Free Languages*. Teubner Studienbucher: Stuttgart, 1979.

[3] C. Cortes, P. Haffner, and M. Mohri. Lattice Kernels for Spoken Dialog Classification. In *Proceedings ICASSP'03*, Hong Kong, 2003.

[4] C. Cortes, P. Haffner, and M. Mohri. Positive Definite Rational Kernels. In *Submitted to The Sixteenth Annual Conference on Computational Learning Theory (COLT 2003)*, 2003.

[5] C. Cortes, P. Haffner, and M. Mohri. Rational Kernels. In *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, Vancouver, Canada, March 2003. MIT Press.

[6] C. Cortes and M. Mohri. Context-Free Recognition with Weighted Automata. *Grammars*, 3(2-3), 2000.

[7] S. Eilenberg. *Automata, Languages and Machines*, volume A-B. Academic Press, 1974.

[8] A. L. Gorin, G. Riccardi, and J. H. Wright. Automated natural spoken dialog. *IEEE Computer Magazine*, 35(4):51–56, April 2002.

[9] P. Haffner, G. Tur, and J. Wright. Optimizing SVMs for complex Call Classification. In *Proceedings ICASSP'03*, 2003.

[10] D. Haussler. Convolution Kernels on Discrete Structures. Technical Report UCSC-CRL-99-10, University of California at Santa Cruz, 1999.

[11] W. Kuich and A. Salomaa. *Semirings, Automata, Languages*. Number 5 in EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berlin, Germany, 1986.

[12] C. Leslie, E. Eskin, J. Weston, and W. S. Noble. Mismatch String Kernels for SVM Protein Classification. In *Advances in Neural Information Processing Systems 15 (NIPS 2002)*, Vancouver, Canada, March 2003. MIT Press.

[13] H. Lodhi, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13 (NIPS 2000)*, pages 563–569. MIT Press, 2001.

[14] M. Mohri. Semiring Frameworks and Algorithms for Shortest-Distance Problems. *Journal of Automata, Languages and Combinatorics*, 7(3):321–350, 2002.

[15] M. Mohri, F. C. N. Pereira, and M. Riley. Weighted automata in text and speech processing. In *ECAI-96 Workshop, Budapest, Hungary*. ECAI, 1996.

[16] Mohri, Mehryar and Fernando C. N. Pereira and Michael Riley. General-purpose Finite-State Machine Software Tools. *http://www.research.att.com/sw/tools/fsm*, AT&T Labs – Research, 1997.

[17] F. C. N. Pereira and M. D. Riley. Speech recognition by composition of weighted finite automata. In E. Roche and Y. Schabes, editors, *Finite-State Language Processing*, pages 431–453. MIT Press, Cambridge, Massachusetts, 1997.

[18] A. Salomaa and M. Soittola. *Automata-Theoretic Aspects of Formal Power Series*. Springer-Verlag: New York, 1978.

[19] R. Schapire, M. Rochery, M. Rahim, and N. Gupta. Incorporating prior knowledge in boosting. In *Proc. of ICML*, 2002.

[20] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press: Cambridge, MA, 2002.

[21] V. N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, New-York, 1998.

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