**Journal of VLSI Signal Processing manuscript No.**
(will be inserted by the editor)

# Acoustic Modelling using Continuous Rational Kernels

**Martin Layton**[1], **Mark Gales**[1]

Department of Engineering, University of Cambridge, Trumpington St., Cambridge, CB2 1PZ

Received: date / Revised version: date

**Abstract** Many discriminative classification algorithms are designed for tasks where samples can be represented by fixed-length vectors. However, many examples in the fields of text processing, computational biology and speech recognition are best represented as variable-length sequences of vectors. Although several dynamic kernels have been proposed for mapping sequences of discrete observations into fixed-dimensional feature-spaces, few kernels exist for sequences of continuous observations. This paper introduces continuous rational kernels, an extension of standard rational kernels, as a general framework for classifying sequences of continuous observations. In addition to allowing new task-dependent kernels to be defined, continuous rational kernels allow existing continuous dynamic kernels, such as Fisher and generative kernels, to be calculated using standard weighted finite-state transducer algorithms. Preliminary results on both a large vocabulary continuous speech recognition (LVCSR) task and the TIMIT database are presented.

## 1 Introduction

In many applications, such as text processing, computational biology and speech recognition, the objects being studied are not fixed-length vectors, but variable-length sequences of observations. One such task, motivating this work, is speech recognition. In its simplest form, this involves classifying sequences of continuous observations into discrete classes (words/phones). Traditionally, this has been performed using generative models and Bayes decision rule. In recent years, however, interest has focused upon discriminative techniques. For speech recognition, training criteria—such as maximum mutual information (MMI) and minimum phone error (MPE)—are often used to estimate parameters for standard generative models [1]. Alternatively, kernel methods, such as support vector machines [2], can be used. These use a kernel mapping function to efficiently transform input examples into a high-dimensional feature-space, suitable for classification. Examples of kernels that map variable-length sequences of observations into a fixed-dimensional feature-space are: string kernels [3], marginalised count kernels [4], Fisher kernels [5] and generative kernels [6]. Whereas string kernels and marginalised count kernels are restricted to sequences of discrete observations, both Fisher and generative kernels can be applied to sequences of either discrete or continuous observations.

Recently, rational kernels [7] were proposed as a general technique for defining and calculating kernel feature-spaces for variable-length sequences of discrete observations. Unlike many other kernels, rational kernels do not prescribe a feature-space mapping to use. Instead, they offer a general framework for defining and calculating feature-spaces using weighted finite-state transducers. Through careful selection of transducers, many standard string and marginalised count kernels can be calculated within the rational kernel framework [7,8]. This allows calculations to be performed using efficient weighted finite-state transducer composition instead of custom dynamic programming algorithms. Kernels can also be defined on lattices of observations, representing multiple alternative hypothesises, allowing effective classification where observation labels are uncertain. In [8–10], rational kernels were used in the construction of a large vocabulary speech recognition system. Here, HMMs mapped continuous observations into a one-dimensional log-likelihood space—the output of each HMM—and a label identifying the HMM. The labels and scores were then passed to the recogniser. Rational kernels, trained upon the recogniser output, were then used to disambiguate sequences of words [8]. Unfortunately in the system, after the initial (HMM) stage of recognition, all acoustic and HMM state-space information is discarded, potentially limiting performance of later stages.

In this paper, continuous rational kernels are proposed. These extend standard rational kernels to allow sequences and lattices of continuous observations to be

classified. Continuous rational kernels consist of two parts: a latent-variable generative model and a kernel. Sequences of continuous or discrete observations are first processed using the generative model and the most likely sequence(s) of latent states through the model are recorded. Rational kernels are then used to classify these sequences of latent states. For speech recognition, this allows the HMM state-space information to be retained and used as part of classification. Using this framework, many continuous dynamic kernels (including Fisher and generative kernels) can be defined and calculated. For both Fisher and generative kernels, this allows derivative-specific dynamic programming algorithms to be replaced by standard transducer operations—different derivatives are calculated by varying the transducers.

This paper is structured as follows. First, an introduction to finite-state transducers and standard (discrete) rational kernels is given. Continuous rational kernels are then presented as an attractive extension for classifying sequences of continuous observations. Calculation of Fisher and generative kernels within the continuous rational kernel framework are then discussed. Examples of calculating first- and second-order HMM derivatives are given. Preliminary results on both a large vocabulary speech recognition task and the TIMIT database [11] are then discussed.

## 2 Dynamic kernels

Many algorithms for statistical classification are based upon the simplifying assumption that examples are represented by fixed-dimensional vectors. For many applications, however, this is not true. Instead examples often consist of a variable-length sequence of observations, prohibiting direct usage of many standard algorithms. Researchers have therefore concentrated on developing feature-space mappings that are capable of converting sequences of observations into fixed-dimensional vectors suitable for use with standard algorithms. Examples of these mappings are: String kernels [3], marginalised count kernels [4], Fisher kernels [5], generative kernels [6,12].

### *2.1 String kernels*

String kernels [3] are a form of dynamic kernel that operates on sequences (strings) of discrete observations (typically sequences of letters that form words). Different strings are compared by considering the number of shared sub-strings. These sub-strings need not be contiguous and are often weighted according to the degree of contiguity. For example, the sub-string 'c-a-r' is present in both the word 'card' and also the word 'custard', but with a lower weight. Although, for many tasks, enumerating all possible sub-strings yields a very high dimensional feature-space, this feature-space is sparse with non-zero entries being calculated using efficient dynamic programming algorithms. Many extensions to this basic concept have been researched, however all suffer from the same problem: they are all restricted to sequences of discrete observations. When continuous observations are used other techniques must be used.

### *2.2 Fisher and generative kernels*

One of the earliest dynamic kernels to embed generative models within a kernel-based framework was the Fisher kernel [5]. This combines the generative (base) model's ability to process variable-length sequences with the kernel machine's flexibility and generalisation performance. Given a set of observation sequences, $\mathcal{O} = \{\boldsymbol{O}_1, \ldots, \boldsymbol{O}_n\}$, $\boldsymbol{O}_i = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_{T_i}\}$, parameters of a single generative model, $\hat{p}(\boldsymbol{O}; \boldsymbol{\lambda})$, are first estimated. The output of this model is a one-dimensional log-likelihood. To capture the differences in generative process between different examples, each example is then mapped to the log-likelihood gradient-space. This is known as the Fisher score-space,

$$\boldsymbol{\phi}^{\mathrm{F}}(\boldsymbol{O}; \boldsymbol{\lambda}) = \nabla_{\boldsymbol{\lambda}} \ln \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}) \tag{1}$$

The resulting kernels (normalised by the Fisher Information matrix) have yielded good performance on some tasks [5,13]. However, when mixture models are used for the base model, 'wrap-around' can occur [6]. This arises when multiple points in the observation-space map to the same point in the score-space, resulting in increased confusion. To minimise this, Smith and Gales proposed generative kernels. Instead of utilising a single shared model, these use separate base models for each class, $\hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}^{(1)})$ and $\hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}^{(2)})$. To further reduce confusion, the log-likelihood ratio of the two classes is also included as a feature. The generative score-space is thus given by,

$$\boldsymbol{\phi}^{\mathrm{LL}}(\boldsymbol{O}; \boldsymbol{\lambda}) = \begin{bmatrix} \ln \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}^{(1)}) - \ln \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}^{(2)}) \\ \nabla_{\boldsymbol{\lambda}^{(1)}} \ln \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}^{(1)}) \\ -\nabla_{\boldsymbol{\lambda}^{(2)}} \ln \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}^{(2)}) \end{bmatrix} \tag{2}$$

Note that Fisher score-spaces are a special case of generative score-spaces where the base model parameters $\boldsymbol{\lambda}^{(1)}$ and $\boldsymbol{\lambda}^{(2)}$ are tied.

### *2.3 Marginalised count kernels*

In [4] Tsuda *et al.* proposed marginalised kernels, another method of combining generative models with traditional kernel classification techniques. Similarly to Fisher kernels, a latent-variable generative model, $\hat{p}(\boldsymbol{O}; \boldsymbol{\lambda})$, is first estimated on the training examples. Next, for each observation $\boldsymbol{O}_i$, instead of extracting features from the generative model, marginalised kernels extract only the posterior probability of latent-state sequence, $P(\boldsymbol{\theta}_i | \boldsymbol{O}_i; \boldsymbol{\lambda})$. These state posterior probabilities are then used as weights

for a second kernel (which must be able to process variable-length sequences). Marginalised kernels are thus defined by,

$$\boldsymbol{K}^{\texttt{Mar}}(\boldsymbol{O}_i, \boldsymbol{O}_j; \boldsymbol{\lambda}) = \sum_{\boldsymbol{\theta}_i \in \boldsymbol{\Theta}} \sum_{\boldsymbol{\theta}_j \in \boldsymbol{\Theta}} P(\boldsymbol{\theta}_i|\boldsymbol{O}_i;\boldsymbol{\lambda}) P(\boldsymbol{\theta}_j|\boldsymbol{O}_j;\boldsymbol{\lambda}) \boldsymbol{K}(\{\boldsymbol{O}_i, \boldsymbol{\theta}_i\}, \{\boldsymbol{O}_j, \boldsymbol{\theta}_j\}) \quad (3)$$

where $\boldsymbol{\theta}_i$ denotes the state sequence associated with an observation $\boldsymbol{O}_i$ and $\boldsymbol{K}(\{\boldsymbol{O}_i, \boldsymbol{\theta}_i\}, \{\boldsymbol{O}_j, \boldsymbol{\theta}_j\})$ is a kernel that measures distances between the points $\{\boldsymbol{O}_i, \boldsymbol{\theta}_i\}$ and $\{\boldsymbol{O}_j, \boldsymbol{\theta}_j\}$.

The inclusion of latent-state posteriors in the kernel yields feature-spaces that resemble first-order generative score-spaces. However, whereas Fisher and generative score-spaces explicitly define the feature-space, marginalised kernels require a second dynamic kernel to perform the actual mapping from a variable-length sequence to a fixed-dimensional feature-space. Unfortunately, this approach suffers from two major disadvantages. First, unlike generative kernels, there is no indication as to which features should be extracted from a sequence. Second, the computational cost of summing over all possible state sequences severely restricts the set of kernels $\boldsymbol{K}(\cdot,\cdot)$ that can be used. Only kernels that simplify the summation in (3) are possible. One such kernel is the count kernel. First- and second-order marginalised count kernels are defined using the feature-spaces $\boldsymbol{\phi}^{\texttt{MC1}}(\boldsymbol{O})$ and $\boldsymbol{\phi}^{\texttt{MC2}}(\boldsymbol{O})$ respectively [4]. Elements of these are given by,

$$\phi^{\texttt{MC1}}_j(\boldsymbol{O}) = \sum_{t=1}^{T} P(\theta^j_t|\boldsymbol{O}) \quad (4a)$$

$$\phi^{\texttt{MC2}}_{jk}(\boldsymbol{O}) = \sum_{t=1}^{T-1} P(\theta^j_t, \theta^k_{t+1}|\boldsymbol{O}) \quad (4b)$$

where $\theta^j_t$ denotes $\theta_t = j$. Marginalised count kernel feature-spaces are related to generative score-spaces. In particular, the first- and second-order feature-spaces are very similar to the score-spaces obtained by differentiating an HMM with respect to its mixture-component priors, $c_m$, and transition probabilities, $a_{ij}$,

$$\nabla_{c_{jm}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{t=1}^{T} \frac{P(\theta^{jm}_t|\boldsymbol{O};\boldsymbol{\lambda})}{c_{jm}} - P(\theta^j_t|\boldsymbol{O};\boldsymbol{\lambda}) \quad (5a)$$

$$\nabla_{a_{ij}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{t=1}^{T-1} \frac{P(\theta^i_t, \theta^j_{t+1}|\boldsymbol{O};\boldsymbol{\lambda})}{a_{ij}} - P(\theta^j_t|\boldsymbol{O};\boldsymbol{\lambda}) \quad (5b)$$

Equations (4a) and (4b) and equations (5a) and (5b) yield functionally similar feature-spaces (but with different scaling and origins). Note that the term $P(\theta^j_t|\boldsymbol{O};\boldsymbol{\lambda})$ in both (5a) and (5b) arises from the Lagrange multipliers used to enforce the sum-to-one constraint on $c_{jm}$ and $a_{ij}$ respectively. Whereas dependency selection in count kernels is ad-hoc, the method of feature extraction that underlies generative kernels offers a systematic method for justifying such dependencies. In addition, although marginalised count kernels are restricted to modelling linear chain dependencies in sequences of discrete observations, $\phi^{\texttt{MC}}_{ij\ldots k}(\boldsymbol{O}) = \sum_{t=1}^{T-n} P(\theta^i_t, \theta^j_{t+1}, \ldots, \theta^k_{t+n}|\boldsymbol{O})$, generative kernels offer a natural framework for modelling complex non-contiguous dependencies between continuous observations.

## 3 Rational kernels

Rational kernels [8,7] are another type of dynamic kernel. Unlike the kernels discussed above, however, rational kernels do not prescribe a feature-space to use. Instead, they provide an easy-to-use framework that allows task-dependent feature-spaces to be defined. This framework is based around weighted finite-state transducers. With careful transducer selection, many of the discrete-observation kernels from the previous section can be calculated within the rational kernel framework.

### *3.1 Weighted finite-state transducers*

Weighted finite-state transducers [14] are a deterministic approach for converting sequences of input symbols into new (often more useful) sequences of output symbols. An example transducer is shown in figure 1. This transducer has two states and five arcs. In general, however, transducers can contain any number of distinct states, labelled from 1 to $N$. States are connected by directed arcs, labelled in the form, $\delta:\gamma$, where $\delta$ and $\gamma$ are the input and output symbols. These are selected from the sets, $\delta \in \{\Sigma \cup \epsilon\}$ and $\gamma \in \{\Delta \cup \epsilon\}$, where $\Sigma$ and $\Delta$ are the input and output alphabets respectively. The null symbol, $\epsilon$, denotes a transition that either does not consume a symbol from the input or does not output a symbol. In later sections, to simplify transducer definitions, some transitions have been grouped. These are labelled as $\Delta:\Delta$ or $\Delta:\epsilon$ and denote the sets of transitions $\{\delta:\delta,\ \forall\delta \in \Delta\}$ and $\{\delta:\epsilon,\ \forall\delta \in \Delta\}$ respectively. This notation assumes that the input and output alphabets are identical.

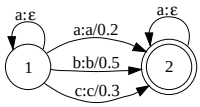



**Fig. 1** An example transducer

The combination of states and arcs allows paths through transducers to be defined. Each path starts at state one[1]

[1] Since state numbering is arbitrary, the states of a transducer with starting state $s \neq 1$ can be simply renumbered so that the $s = 1$.

and terminates in a designated end-state (denoted by a double circle). These paths represent the transformation of an input sequence into a new output sequence. When cyclic transducers are used, such as in figure 1, the one-to-one mapping between input and output sequences is broken. For example, for the transducer in figure 1, both of the input sequences $\{b\}$ and $\{a, a, b, a\}$ map to the same output sequence, $\{b\}$.

In addition to input and output symbols, arcs may be assigned a weight, $w \in \mathbb{K}$, where $\mathbb{K}$ is the set of all valid weights. Arc weights are specified using the notation $\delta : \gamma / w$ where $\delta$, $\gamma$ and $w$ are the input symbol, output symbol and weight respectively. In the absence of a specified weight, arcs are assigned a default weight of $\bar{1}$ where $\bar{1}$ is defined by the transducer semi-ring. This semi-ring defines the minimum set of basic operations required for propagating arc weights through a transducer [15] and is written as $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ [15] where $\oplus$ and $\otimes$ denote operations of addition and multiplication respectively. The zero value, $\bar{0}$, and the identity, $\bar{1}$, are selected to satisfy the identity axioms of addition, $x \oplus \bar{0} = x$, and multiplication, $x \otimes \bar{1} = x$. Some standard semirings are the Real, Log and Tropical semirings (Table 1).

**Table 1** Popular semi-rings for transducers

| Semi-ring | $\mathbb{K}$ | $\oplus$ | $\otimes$ | $\bar{0}$ | $\bar{1}$ |
|---|---|---|---|---|---|
| Real | $\mathbb{R}^+$ | $+$ | $\times$ | 0 | 1 |
| Log | $\mathbb{R} \cup \{\pm\infty\}$ | $\oplus_{\log}$ | $+$ | $+\infty$ | 0 |
| Tropical | $\mathbb{R} \cup \{\pm\infty\}$ | min | $+$ | $+\infty$ | 0 |

where $x \oplus_{\log} y = -\log(\exp(-x) + \exp(-y))$

For transducers, sequences and lattices, weights in the real semiring represent probabilities: they are multiplied along paths and summed when paths merge. When arcs are assigned small weights, however, calculations performed in the real semi-ring can underflow due to limited machine precision. To avoid this, log-probabilities are often used. The log semiring, an isomorphism of the real semiring, allows these log-probabilities to be propagated through transducers and lattices. An extension of this, the tropical semi-ring, enables efficient calculation for large transducers by applying the Viterbi approximation (only the most likely path is considered). For clarity, all transducers in this paper are defined in the real semiring.

A special case of the general transducer is the acceptor. In contrast to transducers which label arcs with both an input and an output symbol, acceptor arcs have only a single symbol, $\delta$. This makes them a natural method of representing sequences or lattices of discrete observations. For example, the sequence $\{a, a, b, a\}$ discussed earlier can be written as in Figure 2. Depending on context, acceptor arc labels can act as either input or output symbols. Similarly to transducers each arc also has an

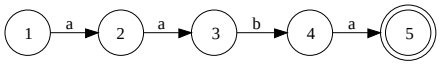



**Fig. 2** An acceptor corresponding to the input sequence $\{a, a, b, a\}$

assigned weight $w$, specified using the notation, $\delta / w$. Arc weights default to $\bar{1}$ unless otherwise specified.

The advantage of representing sequences as acceptors and sequence transformations as transducers is that many useful calculations can be performed using a small number of standard, and efficient, transducer algorithms. Examples of these algorithms are: inversion, composition [9] and shortest-distance [15] (transducer weight). The inverse, $\boldsymbol{U}_1^{-1}$, of a general transducer, $\boldsymbol{U}_1$, is calculated by transposing input and output symbols along all transducer arcs. Transducer composition allows complex transducers to be constructed from simpler component parts. The composition of two transducers, $\boldsymbol{U}_1$ and $\boldsymbol{U}_2$, is written as $\boldsymbol{U}_1 \circ \boldsymbol{U}_2$ and is defined as the transducer that, given any input sequence, generates an output sequence equivalent to that generated from passing the input through $\boldsymbol{U}_1$ and the output of that through $\boldsymbol{U}_2$. The final operator, weighted transducer shortest distance, calculates the total weight from all possible paths through the transducer [15]. Using these operations, rational kernels can be defined.

### *3.2 Rational kernels*

Using the weighted finite-state transducer framework discussed above, rational kernels [8] can be defined. These allow high-dimensional kernel feature-spaces to be defined and calculated using only simple transducer operations on discrete sequences (or lattices) of observations. The efficiency of this mapping arises from the fact that distances in the feature-space can be calculated using standard transducer operations on the input sequences/lattices. It is not necessary to explicitly calculated the feature-space.

Consider a simple example. Let $\boldsymbol{O} = \{o_1, \ldots, o_T\}$ be a sequence, of length $T$, with discrete observations $o_t$ selected from the input alphabet $\Sigma$. Since many discriminative classification algorithms can only operate on fixed-dimensional vectors, a mapping from this variable-length sequence to a fixed-dimensional vector is desired. One such mapping is the feature-space defined by the counts of the occurrences of each symbol in the input alphabet. This is similar to a Bag-of-Words kernel [16] and is known as the unigram feature-space. In vector notation it is written as,

$$\boldsymbol{\phi}(\boldsymbol{O}) = \begin{bmatrix} f(a|\boldsymbol{O}) \\ f(b|\boldsymbol{O}) \\ \vdots \end{bmatrix} \tag{6}$$

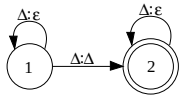
where $f(\delta|\boldsymbol{O})$ represents the number of occurrences of the symbol $\delta$ in $\boldsymbol{O}$. The distance between examples in the feature-space is given by the feature-space dot-product. This is known as the unigram kernel. For two examples, $\boldsymbol{O}_i$ and $\boldsymbol{O}_j$, this is written as,

$$\begin{aligned}\boldsymbol{K}(\boldsymbol{O}_i, \boldsymbol{O}_j) &= \boldsymbol{\phi}(\boldsymbol{O}_i)^T \boldsymbol{\phi}(\boldsymbol{O}_j) \\ &= \sum_{\delta \in \Delta} f(\delta|\boldsymbol{O}_i) f(\delta|\boldsymbol{O}_j) \end{aligned} \quad (7)$$

In the above example, the unigram feature-space is calculated explicitly. The computational cost of calculating the kernel is therefore proportional to the dimensionality of the feature-space (the number of elements in the input alphabet). When large alphabets are used (e.g. words in a large vocabulary speech recognition system), the finite-state transducer framework in section 3.1 allows an alternative approach.

First, acceptors $\boldsymbol{A}_i$ are constructed to represent the observation sequences $\boldsymbol{O}_i$; these have a structure similar to that in Figure 2. Next, the unigram transducer, $\boldsymbol{U}$, is introduced,

$\boldsymbol{U}$ =




When an acceptor $\boldsymbol{A}_i$, representing the example $\boldsymbol{O}_i$, is composed with this transducer, a lattice, $\boldsymbol{A}_i \circ \boldsymbol{U}$, is generated. This lattice contains $T_i$ distinct paths of length $T_i$ (where $T_i$ is the length of the observation sequence). All output symbols in the $t$-th, $t \in [1, T_i]$, path are null except for the transition associated with $o_t$ which has an output label $o_t \in \Delta$. The number of paths with a particular output label, $\delta \in \Delta$, is equal to the number of occurrences of $\delta$ in the sequence. Since each path has a weight of 1.0, the total weight of these paths is $f(\delta|\boldsymbol{O})$—the unigram count.

In lattice form, these unigram features are difficult to use. However, using transducer composition and shortest distance calculation, the dot-product of these lattice-based feature-spaces can be calculated. This feature-space dot-product operation is known as the unigram rational kernel [8] and is written as,

$$\begin{aligned}\boldsymbol{K}(\boldsymbol{O}_i, \boldsymbol{O}_j) &= [[(\boldsymbol{A}_i \circ \boldsymbol{U}) \circ (\boldsymbol{A}_j \circ \boldsymbol{U})^{-1}]] \\ &= [[\boldsymbol{A}_i \circ \boldsymbol{U} \circ \boldsymbol{U}^{-1} \circ \boldsymbol{A}_j]] \end{aligned} \quad (8)$$

Inversion of the second operand transposes the input and output symbols. The outputs of the first operand therefore join with the inputs of the second, matching feature-space dimensions.

Unfortunately the unigram kernel is not invariant to sequence lengths: longer sequences have more observations and therefore higher counts associated with each dimension. Sequence-length normalisation is therefore preferable; the length-normalised unigram kernel is given by,

$$\boldsymbol{K}(\boldsymbol{O}_i, \boldsymbol{O}_j) = \frac{1}{T_i T_j} [[\boldsymbol{A}_i \circ \boldsymbol{U} \circ \boldsymbol{U}^{-1} \circ \boldsymbol{A}_j]] \quad (9)$$

In addition to basic sequences of observations, rational kernels can operate on lattices.When labelling of observations is uncertain, the use of lattices allows all possible sequences of labels to be represented.[2] Each path in the lattice represents a single hypothesised label sequence and is assigned a weight according to its likelihood. When the kernel is calculated, all information in the lattice is utilised, potentially improving classification performance. The rational kernel for lattices is identical to that for linear sequences and so can be calculated using only standard transducer operations.

Unfortunately, rational kernels can only be defined for sequences of discrete observations whereas many practical tasks require classification of sequences of *continuous* observations. A method of processing these sequences is required. In this paper, continuous rational kernels are proposed as a powerful extension of rational kernels that enables sequences of continuous observations to be classified.

## 4 Continuous Rational Kernels

As discussed in the previous section, finite-state automata provide an attractive framework for defining kernels on variable-length sequences of *discrete* data. Unfortunately many tasks require classification of sequences of *continuous* observations. A method of mapping variable-length sequences of continuous observations to a fixed-dimensional feature-space is therefore required.

Continuous rational kernels, proposed in this paper, offer an attractive solution. Similarly to Fisher and generative kernels, continuous rational kernels are defined using a combination of two different approaches: generative models and transducer-based rational kernels. This combination allows them to define complex, task-dependent, metrics for sequences of continuous observations. Similarly to generative kernels, latent-variable class-conditional base models must be defined. Base model parameters are typically estimated using either ML [17] or MMI [18] estimation. For clarity, the discussion in this section will only consider derivatives of a single base model.

Continuous rational kernels consist of two separate stages. The first stage converts sequences of continuous observations into a corresponding sequence of discrete observations. This conversion is achieved by recording the latent-state sequences (corresponding to the observation sequences) through the base model. Viterbi decoding yields a single sequence (the most likely) whereas

[2] To reduce memory and CPU requirements, lattices are often pruned to remove highly unlikely paths.

Forwards-Backwards decoding yields multiple sequences (corresponding to different state alignments). Both can be compactly represented using a weighted finite-state acceptor, $\boldsymbol{L}$, with arcs labelled with probabilities and state/mixture-component pairs. When calculating base model derivatives, Forwards-Backwards decoding must be used (the resulting acceptor has an identical structure to the standard HMM 'trellis diagram' [19]). Next, given these weighted acceptors (with discrete labels), rational kernels can be defined. We call them continuous rational kernels. The kernel feature-space is defined by a transducer operating on the latent-state acceptor, $\boldsymbol{L}$.

In this paper, $n$-gram and gappy-$n$-gram transducers are described. In addition to allowing many forms of string kernel to be defined (for discrete observations), these allow many different latent-state-posterior probabilities to be calculated. With appropriate weighting, this allows both first and higher-order derivatives of the base acoustic models with respect to their parameters to be computed within the continuous rational kernel framework. Unlike direct calculation of generative kernels, custom dynamic programming algorithms are not required.

### *4.1 Component and transition probability kernels*

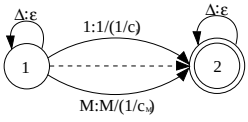
First consider an $M$-component GMM base acoustic model with parameters $\boldsymbol{\lambda}$. Derivatives of the GMM log-likelihood with respect to its mixture-component priors, $c_m \subset \boldsymbol{\lambda}$, are given by,

$$\begin{aligned}\boldsymbol{\phi}^{\mathsf{c}}_m(\boldsymbol{O};\boldsymbol{\lambda}) &= \nabla_{c_m} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) \\ &= \sum_{t=1}^{T} \left[ \frac{P(m|\boldsymbol{o}_t;\boldsymbol{\lambda})}{c_m} - 1 \right] \end{aligned} \tag{10}$$

where $\boldsymbol{\phi}^{\mathsf{c}}_m$ denotes the $m$-th element of the score-space (corresponding to the derivative with respect to $c_m$). It is interesting to compare this component-prior score-space with the feature-space generated using the unigram transducer, $\boldsymbol{U}$, described in section 3.2. Applying $\boldsymbol{U}$ to the lattice of latent-states, $\boldsymbol{L}$ (corresponding to the observation sequence $\boldsymbol{O}$), yields a feature-space with elements given by the posterior probabilities of the GMM mixture-components. For continuous rational kernels, this unigram feature-space is given by,

$$\boldsymbol{\phi}^{\mathtt{uni}}(\boldsymbol{O};\boldsymbol{\lambda}) = \boldsymbol{L} \circ \boldsymbol{U} = \sum_{t=1}^{T} \begin{bmatrix} P(m_t = 1|\boldsymbol{O};\boldsymbol{\lambda}) \\ P(m_t = 2|\boldsymbol{O};\boldsymbol{\lambda}) \\ \cdots \\ P(m_t = M|\boldsymbol{O};\boldsymbol{\lambda}) \end{bmatrix} \tag{11}$$

From equations (10) and (11), it is clear that the mixture-component derivative score-space is simply a scaled version of the unigram feature-space (the additional constant in equation (10) can be ignored since it does not affect classification). The GMM score-space can thus be calculated using a scaled unigram transducer, $\boldsymbol{U}^{/\mathsf{c}}_{\mathtt{gmm}}$,

$\boldsymbol{U}^{/\mathsf{c}}_{\mathtt{gmm}} =$ 


The corresponding component-prior continuous rational kernel for GMMs (after sequence length-normalisation) is therefore given by,

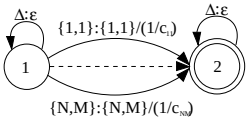
$$\boldsymbol{K}^{\mathsf{c}}(\boldsymbol{O}_i, \boldsymbol{O}_j;\boldsymbol{\lambda}) = \frac{1}{T_i T_j}[[\boldsymbol{L}_i \circ \boldsymbol{U}^{/\mathsf{c}}_{\mathtt{gmm}} \circ \boldsymbol{U}^{/\mathsf{c}^{-1}}_{\mathtt{gmm}} \circ \boldsymbol{L}_j]] \tag{12}$$

and is expressed entirely in terms of transducer operations, allowing standard algorithms to be used.

Using this framework, more complex derivatives can be calculated. Consider, for example, derivatives of an $N$-component, $M$-mixture-component HMM with respect to the mixture-component priors,

$$\begin{aligned}\boldsymbol{\phi}^{\mathsf{c}}_{jm}(\boldsymbol{O};\boldsymbol{\lambda}) &= \nabla_{c_{jm}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) \\ &= \frac{1}{c_{jm}} \sum_{t=1}^{T} P(\theta_t = \{j,m\}|\boldsymbol{O};\boldsymbol{\lambda}) - \sum_{t=1}^{T} P(\theta_t = s_j|\boldsymbol{O};\boldsymbol{\lambda}) \end{aligned} \tag{13}$$

where $\boldsymbol{\phi}^{\mathsf{c}}_{jm}$ denotes the element of the score-space corresponding to the derivatives with respect to the HMM mixture-component priors. Here, feature-space elements consist of two terms. The first is the scaled posterior probabilities of state/mixture-component pairings. This is similar to the feature-space in equation (11) and can be calculated using a scaled unigram transducer, $\boldsymbol{U}^{/\mathsf{c}}_{\mathtt{hmm}}$,

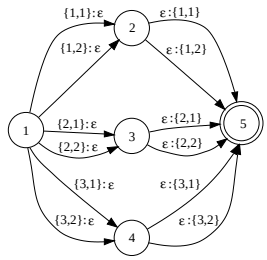
$\boldsymbol{U}^{/\mathsf{c}}_{\mathtt{hmm}} =$ 


The second term in equation (13) is more complex to calculate since the input acceptor (corresponding to the observation sequence) has no concept of what $P(\theta_t = s_j|\boldsymbol{O};\boldsymbol{\lambda})$ means. Instead, the state-posterior must be expanded in terms of the posterior probabilities of state/mixture-component pairs,

$$P(\theta_t = s_j|\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{m=1}^{M} P(\theta_t = \{j,m\}|\boldsymbol{O};\boldsymbol{\lambda}) \tag{14}$$

This yields a summation of the state/mixture-component posteriors over the components in each state. Given the unigram feature-space, a transducer, $\boldsymbol{S}$, can be defined to perform this summation. An example of $\boldsymbol{S}$ for a three-state HMM with two mixture-components is,

$\boldsymbol{S}$ =




The state posterior, $P(\theta_t = s_j|\boldsymbol{O};\boldsymbol{\lambda})$, is therefore the composition of the two transducers: $\boldsymbol{U}_{\texttt{hmm}} \circ \boldsymbol{S}$, where $\boldsymbol{U}_{\texttt{hmm}}$ is the unscaled version of $\boldsymbol{U}'^{\texttt{c}}_{\texttt{hmm}}$. The resulting component-prior kernel for an HMM is given by,

$$\begin{aligned}\boldsymbol{K}^{\texttt{c}}_{\texttt{hmm}}(\boldsymbol{O}_i,\boldsymbol{O}_j;\boldsymbol{\lambda}) &= \frac{1}{T_i T_j}\Big([[\boldsymbol{L}_i \circ \boldsymbol{U}'^{\texttt{c}}_{\texttt{hmm}} \circ \boldsymbol{U}'^{\texttt{c}^{-1}}_{\texttt{hmm}} \circ \boldsymbol{L}_j]] \\ &\quad - 2[[\boldsymbol{L}_i \circ \boldsymbol{U}'^{\texttt{c}}_{\texttt{hmm}} \circ \boldsymbol{S}^{-1} \circ \boldsymbol{U}^{-1}_{\texttt{hmm}} \circ \boldsymbol{L}_j]] \\ &\quad + [[\boldsymbol{L}_i \circ \boldsymbol{U}_{\texttt{hmm}} \circ \boldsymbol{S} \circ \boldsymbol{S}^{-1} \circ \boldsymbol{U}^{-1}_{\texttt{hmm}} \circ \boldsymbol{L}_j]]\Big)\end{aligned} \tag{15}$$

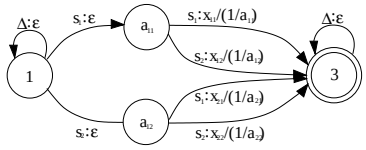
This kernel uses exactly the same framework and calculations as the component-prior kernel for GMMs. The only change required is the addition of a new transducer, $\boldsymbol{S}$, to calculate the state posteriors.

For HMM base models, derivatives with respect to the transition probabilities can be calculated. These are given by,

$$\begin{aligned}\nabla_{a_{ij}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) &= \frac{1}{a_{ij}} \sum_{t=1}^{T} P(\theta_t = s_i, \theta_{t+1} = s_j|\boldsymbol{O};\boldsymbol{\lambda}) \\ &\quad - \sum_{t=1}^{T} P(\theta_t = s_j|\boldsymbol{O};\boldsymbol{\lambda})\end{aligned} \tag{16}$$

The first term can be calculated using a modified form of a bigram feature-space, calculated using the transducer, $\boldsymbol{B}^{/a}$. An example of $\boldsymbol{B}^{/a}$ for a two-state HMM is,

$\boldsymbol{B}^{/a}$ =




where input labels $s_i$ represent the set of all transitions with a state $i$. Dimensions in the feature-space are indexed by pairs of consecutive states, $\{i,j\}$, and labelled $x_{ij}$. The continuous rational kernel for the transition probability score-space can therefore be calculated using an expression similar to (15).

### *4.2 Fisher and generative score-spaces*

In the previous section, the use of modified unigram and bigram transducers for the calculation of derivatives of the base acoustic model with respect to the mixture-component priors and transition probabilities was discussed. However, score-spaces can also be defined using the derivatives with respect to the mixture-component means, $\boldsymbol{\mu}_{jm}$, and covariances, $\boldsymbol{\Sigma}_{jm}$. Consider the score-space of derivatives with respect to the means [6],[3]

$$\begin{aligned}\boldsymbol{\phi}^{\texttt{m}}_{jm}(\boldsymbol{O};\boldsymbol{\lambda}) &= \nabla_{\boldsymbol{\mu}_{jm}} \ln \hat{p}(\boldsymbol{O};\boldsymbol{\lambda}) \\ &= \sum_{t=1}^{T} P(\theta_t = \{j,m\}|\boldsymbol{O};\boldsymbol{\lambda}) \boldsymbol{\Sigma}^{-1}_{jm}(\boldsymbol{o}_t - \boldsymbol{\mu}_{jm})\end{aligned} \tag{17}$$

where $\boldsymbol{\phi}^{\texttt{m}}_{jm}$ denotes elements of the score-space. Although, given an acceptor of state-sequences, unigram transducers generate state/component posteriors, they cannot weight them by the observations. This is because observations are vector quantities whereas transducer weights are scalar. To overcome this, a new semiring, the *vector semiring*, is introduced to allow weight *vectors* to be associated with transducer transitions. It is defined as $(\mathbb{R}^{d^+}, +, \otimes_{\text{vec}}, [0]^d, [1]^d)$ where $[x]^d$ represents a vector of length $d$ with all elements equal to $x$. Both addition and multiplication are performed on a per-dimension basis; multiplication is defined as, $x \otimes_{\text{vec}} y = \{x_1 y_1, x_2 y_2, \ldots, x_d y_d\}$. For easy comparison with the standard semi-rings (Table 1), this information is summarised in Table 2.

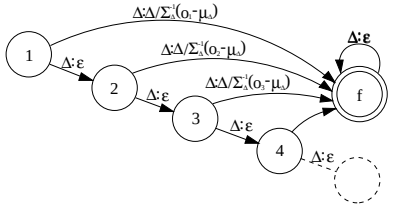
**Table 2** The Vector semiring

| Semi-ring | $\mathbb{K}$ | $\oplus$ | $\otimes$ | $\bar{0}$ | $\bar{1}$ |
|---|---|---|---|---|---|
| Vector | $\mathbb{R}^{d^+}$ | $+$ | $\otimes_{\text{vec}}$ | $\{0\}^d$ | $\{1\}^d$ |

where $x \otimes_{\text{vec}} y = \{x_1 y_1, x_2 y_2, \ldots, x_d y_d\}$

The second problem with weighting posteriors by observations is that observations are time-dependent, whereas transducer weights must be constant. One solution is to expand the self-transitions in the unigram transducer to create time-dependent transducer paths. These time-dependent paths are then be weighted by a function of the $t$-th observation, $\boldsymbol{\Sigma}^{-1}_{jm}(\boldsymbol{o}_t - \boldsymbol{\mu}_{jm})$. The expanded unigram transducer with vector weights is given by,

$\boldsymbol{F}$ =




This is known as the Fisher transducer. The Fisher kernel with acoustic model derivatives with respect to the

[3] Covariance derivatives have a similar functional form to derivatives with respect to the mean. For brevity, they are omitted.

means is given by,

$$\boldsymbol{K}^{\mathtt{m}}(\boldsymbol{O}_i, \boldsymbol{O}_j; \boldsymbol{\lambda}) = \frac{1}{T_i T_j}[[\boldsymbol{L}_i \circ \boldsymbol{F} \circ \boldsymbol{F}^{-1} \circ \boldsymbol{L}_j]] \quad (18)$$

Covariance derivatives are obtained simply by setting the Fisher transducer path weights to $-\frac{1}{2}[\boldsymbol{\Sigma}_{jm}^{-1} + \boldsymbol{\Sigma}_{jm}^{-1}(\boldsymbol{o}_t - \boldsymbol{\mu}_{jm})(\boldsymbol{o}_t - \boldsymbol{\mu}_{jm})^T \boldsymbol{\Sigma}_{jm}^{-1}]$.

### *4.3 Second and higher-order derivatives*

Previous sections have concentrated on the use of continuous rational kernels for the calculation of the first-order derivatives of GMM and HMM acoustic models with respect to their parameters. Additional information, that may be useful for classification, is contained within the higher-order derivatives of the acoustic models. When these higher-order derivatives are considered, rational kernels offer significant benefits over their dynamic programming counterparts. Consider, for example, the second derivatives of an HMM with respect to its component priors,

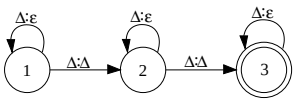
$$\begin{aligned}\nabla_{c_{kn}} \nabla_{c_{jm}}^T \ln \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}) = \quad (19)\\ \frac{1}{c_{jm} c_{kn}} \sum_{t=1}^{T} \sum_{\tau=1}^{T} \Big( D(\theta_t^{jm}, \theta_\tau^{kn}) - c_{jm} D(\theta_t^{j}, \theta_\tau^{kn}) \\ - c_{kn} D(\theta_t^{jm}, \theta_\tau^{k}) + c_{jm} c_{kn} D(\theta_t^{j}, \theta_\tau^{k}) \Big) \\ - \frac{2}{c_{jm} c_{kn}} \sum_{t=1}^{T} P(\theta_t^{jm} | \boldsymbol{O}; \boldsymbol{\lambda}) \delta_{jk} \delta_{mn}\end{aligned}$$

where

$$\begin{aligned}D(\theta_t^{jm}, \theta_\tau^{kn}) = P(\theta_t^{jm}, \theta_\tau^{kn} | \boldsymbol{O}; \boldsymbol{\lambda}) - \quad (20)\\ P(\theta_t^{jm} | \boldsymbol{O}; \boldsymbol{\lambda}) P(\theta_\tau^{kn} | \boldsymbol{O}; \boldsymbol{\lambda})\end{aligned}$$

and $\theta_t^{jm}$ and $\theta_t^{j}$ denote $\theta_t = \{j, m\}$ and $s_t = j$ respectively. Dynamic programming algorithms for calculating the joint posterior of being in state $\{j, m\}$ at time $t$ and state $\{k, n\}$ at time $\tau$ are complex. This makes direct calculation of second derivatives difficult. However, when calculated within the continuous rational kernel framework, standard algorithms are reused with new transducers. In particular, the transducer for calculating the joint probability, $P(\theta_t^{jm}, \theta_\tau^{kn} | \boldsymbol{O}; \boldsymbol{\lambda})$, is given by the gappy bigram transducer, $\boldsymbol{G}$. This generates the feature-space, $\boldsymbol{\phi}_{jm,kn}^{\mathtt{gb}}(\boldsymbol{O}; \boldsymbol{\lambda})$,

$\boldsymbol{G}$ =




$$\begin{aligned}\boldsymbol{\phi}_{jm,kn}^{\mathtt{gb}}(\boldsymbol{O}; \boldsymbol{\lambda}) = \quad (21)\\ \sum_{t=1}^{T} \sum_{\tau=1}^{T} P(\theta_t = \{j, m\}, \theta_\tau = \{k, n\} | \boldsymbol{O}; \boldsymbol{\lambda})\end{aligned}$$

Scaling the arcs of this transducer by $1/c_{jm}c_{kn}$ (similarly to the unigram and bigram transducers) yields a transducer that calculates the first term in equation (20). All other terms can be generated using combinations of the HMM unigram ($\boldsymbol{U}_{\mathtt{hmm}}$), gappy-bigram ($\boldsymbol{G}$), and summation ($\boldsymbol{S}$) transducers. For example, the term, $P(\theta_t^{jm} | \boldsymbol{O}; \boldsymbol{\lambda}) P(\theta_\tau^{kn} | \boldsymbol{O}; \boldsymbol{\lambda})$, is given by the product of two unigram feature-spaces, $(\boldsymbol{L}_i \circ \boldsymbol{U}_{\mathtt{hmm}}) \otimes (\boldsymbol{L}_j \circ \boldsymbol{U}_{\mathtt{hmm}})$, where $\otimes$ denotes transducer concatenation. The second-order component-prior kernel can thus be written as a sum of the kernels of the parts; for brevity it is omitted. In addition to second derivatives with respect to the component priors, second derivatives with respect to other parameters can be calculated. These derivatives take a similar form to equation (19) and may be calculated using the transducers introduced in this section. Higher-order derivatives are calculated using other gappy-$n$-gram transducers, for example, the gappy-trigram transducer for third-derivatives.

### *4.4 Generative score-spaces and kernels*

In the previous sections, continuous rational kernels have been proposed as a systematic method for calculating derivatives of HMM base models with respect to the transition probabilities and mixture-component priors, means and variances. Thus far they have been calculated separately. In practice, however, Fisher and generative score-spaces are formed from a combination of these derivatives. For example, a typical first-order generative score-space is given by,

$$\boldsymbol{\phi}^{\mathtt{wmc}}(\boldsymbol{O}; \boldsymbol{\lambda}) = \begin{bmatrix} \ln \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}^{(1)}) - \ln \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}^{(2)}) \\ \nabla_{\boldsymbol{c}^{(1)}} \ln \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}^{(1)}) \\ \nabla_{\boldsymbol{\mu}^{(1)}} \ln \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}^{(1)}) \\ \nabla_{\boldsymbol{\Sigma}^{(1)}} \ln \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}^{(1)}) \\ -\nabla_{\boldsymbol{c}^{(2)}} \ln \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}^{(2)}) \\ -\nabla_{\boldsymbol{\mu}^{(2)}} \ln \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}^{(2)}) \\ -\nabla_{\boldsymbol{\Sigma}^{(2)}} \ln \hat{p}(\boldsymbol{O}; \boldsymbol{\lambda}^{(2)}) \end{bmatrix} \quad (22)$$

where $\nabla_{\boldsymbol{c}^{(1)}}$, $\nabla_{\boldsymbol{\mu}^{(1)}}$ and $\nabla_{\boldsymbol{\Sigma}^{(1)}}$ denote the vector of derivatives with respect to the mixture-component priors, means and variances respectively. The corresponding kernel,

$$\boldsymbol{K}^{\mathtt{wmc}}(\boldsymbol{O}_i, \boldsymbol{O}_j; \boldsymbol{\lambda}) = \boldsymbol{\phi}^{\mathtt{wmc}}(\boldsymbol{O}_i; \boldsymbol{\lambda})^T \boldsymbol{\phi}^{\mathtt{wmc}}(\boldsymbol{O}_j; \boldsymbol{\lambda}) \quad (23)$$

can be written as a sum of the kernels of the individual parts,

$$\begin{aligned}\boldsymbol{K}^{\mathtt{wmc}}(\boldsymbol{O}_i, \boldsymbol{O}_j; \boldsymbol{\lambda}) = \boldsymbol{K}^{\mathtt{l}}(\boldsymbol{O}_i, \boldsymbol{O}_j; \boldsymbol{\lambda}) + \boldsymbol{K}^{\mathtt{c}}(\boldsymbol{O}_i, \boldsymbol{O}_j; \boldsymbol{\lambda}^{(1)}) \\ + \boldsymbol{K}^{\mathtt{m}}(\boldsymbol{O}_i, \boldsymbol{O}_j; \boldsymbol{\lambda}^{(1)}) + \cdots \quad (24)\end{aligned}$$

where $\boldsymbol{K}^{\mathtt{l}}(\boldsymbol{O}_i, \boldsymbol{O}_j; \boldsymbol{\lambda})$ is the dot-product of the log-likelihood ratios and $\boldsymbol{K}^{\mathtt{c}}(\boldsymbol{O}_i, \boldsymbol{O}_j; \boldsymbol{\lambda}^{(1)})$, $\boldsymbol{K}^{\mathtt{m}}(\boldsymbol{O}_i, \boldsymbol{O}_j; \boldsymbol{\lambda}^{(1)})$, etc., are the individual derivative kernels (calculated using the continuous rational kernel framework and the transducers discussed earlier). Note that the summation of kernels can be performed using the union operator on the individual transducers.

## 5 Experimental Results

Preliminary experiments were performed to examine the use of generative kernels in speech recognition. Data from two different databases were used. The first is a set of confusable words, extracted from the `fsh2004sub` data set[20]. This is used both within a cross-validation framework and with a held-out test set, the `eval03` data set [20]. The second database is a set of phone pairs extracted from the TIMIT phone classification task [11].

### *5.1 LVCSR*

Generative kernels (calculated using either the continuous rational kernel framework or dynamic programming) are a powerful form of acoustic model. Unfortunately, the distance-learning algorithms, in particular SVMs, used to train these models are binary classifiers whereas large vocabulary continuous speech recognition (LVCSR) has a vast number of possible classes. In order to apply continuous rational kernels to LVCSR, it is necessary to map this highly complex problem into a small set of binary classification problems. An approach related to that described in [13] is used.

Given an utterance, a standard LVCSR Viterbi decoder is used to generate a word lattice. This represents the most likely word sequences for that utterance. The arcs are labelled with words and the language and acoustic model likelihoods; nodes are labelled with time-stamps. Next, word lattices are converted to confusion networks [21]. These consist of a series of nodes with a linear graph. Each arc is labelled with a word, start and end times and a log-posterior, $\mathcal{F}(\omega_i)$. Finally the confusion networks are pruned so that at each time instance, only two words remain. Acoustic models and continuous rational kernels are then trained on these word pairs.

The database used for the LVCSR experiments was a 400 hour subset of the Fisher LDC data. This is the `fsh2004sub` data set used for initial system development of the system described in [20]. The model set used in experiments was based upon the standard models and front-end described in [20]. Confusion networks were generated, using a bigram language model (LM), on the same 400 hours of training data.

Classifier performance was evaluated using 8-fold cross-validation on the training data. All experiments used diagonal covariance matrix GMMs, trained on the acoustic data using the longest time-stamps from the confusion networks[4] for the two confusable words. The number of positive and negative examples within each word pair were equalised by sampling – random selection yields an error rate of 50%. A number of pairwise classifiers were then trained; two examples are shown in table 3. The baseline performance of the LVCSR system for each confusable pair is given by the confusion network (CN) posteriors—the baseline error rate is approximately 20%.

**Table 3** 8-Fold cross validation results (% error) using GMMs with a variable number of mixture components

| Word Pair (examples) | Training | CN post. | # Components 1 | 2 | 4 |
|---|---|---|---|---|---|
| CAN/CAN'T (7,522) | ML | 21.5 | 18.3 | 14.0 | 12.0 |
| | SVM $\phi^{\mathtt{w}}$ | | – | 13.6 | 11.1 |
| | SVM $\phi^{\mathtt{mc}}$ | | 16.1 | 12.4 | 10.6 |
| | SVM $\phi^{\mathtt{wmc}}$ | | – | 12.3 | 10.4 |
| | SVM $\phi^{\mathtt{pmc}}$ | | 11.4 | 9.9 | 8.7 |
| | SVM $\phi^{\mathtt{pwmc}}$ | | – | 9.9 | 8.5 |
| KNOW/NO (8,950) | ML | 16.9 | 31.6 | 30.6 | 29.2 |
| | SVM $\phi^{\mathtt{w}}$ | | – | 30.7 | 28.1 |
| | SVM $\phi^{\mathtt{mc}}$ | | 30.7 | 28.2 | 26.4 |
| | SVM $\phi^{\mathtt{wmc}}$ | | – | 28.3 | 26.1 |
| | SVM $\phi^{\mathtt{pmc}}$ | | 15.1 | 14.7 | 13.7 |
| | SVM $\phi^{\mathtt{pwmc}}$ | | – | 14.7 | 13.7 |

Score-space elements are denoted by:
- m Derivatives w.r.t. means
- c Derivatives w.r.t. covariances
- w Derivatives w.r.t. mixture-component priors
- p Confusion network posterior ratio

Standard ML estimated GMMs with one, two and four Gaussian mixture-components were used as baseline pairwise classifiers. Given these, a number of SVMs were trained using generative score-spaces of the log-likelihood ratio plus derivatives. Different combinations of derivatives with respect to the mixture-component priors (w), means (m) and covariances (c) were considered. In almost all cases, SVMs gave performance gains over the ML estimated GMMs. As the number of derivatives increased, performance increased. Best performance was achieved when all derivatives ($\boldsymbol{\phi}^{\mathtt{wmc}}$) were included. Similar trends were observed in [6,22] for the Deterding and Isolet datasets. For the CAN/CAN'T pair, the GMM and SVM systems obtained better results than the baseline confusion network score. However in general, this was not the case.

Classifier accuracy was improved when the confusion network posterior ratio, $\mathcal{F}(\omega_1) - \mathcal{F}(\omega_2)$, was included in the score-space; this added context information from the LVCSR language model. In all cases, SVMs with confusion network posteriors in their score-spaces outperformed both the confusion network baseline and SVMs without the posterior ratio—significant gains were observed for word pairs where both GMM and SVM performance was poor.

Experiments with second- and higher-order score-spaces were not performed due to the high-dimensionality (39-dimension) of the observation-space. This has the effect of polarising the state/mixture-component posteriors to be either one or zero. Since, as shown in equa-

[4] The earliest time of the two words and the latest time of the two words are used for the start and end time respectively. This ensures that the full acoustic data for both words is included.

tions (19) and (20), second derivatives are expressed in terms of the joint posterior of two states minus the product of the posteriors of the two states, they are always zero[5]. In general this is not always the case. When low-dimensional observation-spaces are used, state/mixture-component posteriors assume a range of values, allowing second derivatives to assume non-zero values.

Unfortunately, with ∼65,000 possible words there are vast number ($\sim 2 \times 10^9$) of possible word pairs. This means that training data is scarce, limiting the number of pairwise classifiers that can be trained (to ∼30). This restricts the proportion of any test set that can be rescored, limiting the potential gains. This is easily demonstrated on the 6-hour `eval03` test set [20]. SVMs were trained to disambiguate ten of the most confusable pairs. Of the 1,250 word pairs rescored, 56 corrections were obtained (4.5%). However, in comparison to whole test set (76,157 words), this corresponds to an improvement of $< 0.1\%$. This is too small to consider including pairwise rescoring in any standard LVCSR system. However, for tasks where there are a much smaller number of confusions, allowing a greater proportion of the test data to be rescored, results suggest that reasonable gains can be achieved.

### *5.2 TIMIT*

The TIMIT phone classification task [11] consists of approximately 3 hours of training data, transcribed with phone markings and time stamps. There are 142,910 phone segments, each of which is labelled with one of 48 phone labels. This results in a total of 1,128 possible phone confusion pairs (compared to $\sim 2 \times 10^9$ word pairs for the LVCSR task). With a large number of training examples and few phone pairs it is possible to train robust classifiers to disambiguate a much larger percentage of the test data than was possible for the previous LVCSR task. Rescoring is therefore expected to yield a much larger improvement in the overall error rate.

The acoustic data was coded using the experimental setup described in [23]. Results are reported on the MIT development test set [24] and the NIST core test set. The baseline HMM-based phone classification system was estimated using maximum likelihood estimation and 3-state, 10-mixture-component monophone HMMs. As a classification task (the time alignment is known), pruned confusion networks are trivial to construct: the confusion pair for each example is simply given by the two most likely phones according to the HMM baseline system. Next, pairwise augmented models were trained to disambiguate the twenty most confusable pairs. These used 3-state, 4-mixture-component class-conditional HMMs as base models.

The pairwise phone classifiers showed similar trends to those described in the LVCSR experiments above. Additionally, since confusion networks were generated without the benefit of a language model, almost all augmented models outperformed the CN decoding baseline—the few that underperformed appeared to be over-trained. Without a language model there is little benefit in using the augmented model/confusion network combinations discussed previously (section 5.1). The baseline systems are therefore rescored using only the generative kernels.

Given the set of trained augmented models, performance on the MIT development test set was evaluated. Classifiers were sorted—best first—according to their performance on this task. Cumulative performance is shown in figure 3. Then, using this ordering, performance on the NIST core test set was evaluated. This is also shown in figure 3. From the graph, it is clear that

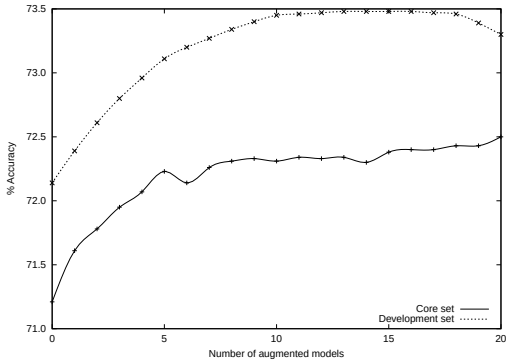



**Fig. 3** Rescoring of an ML baseline for the TIMIT phone classification task

performances on both the development set and the core test set peak at approximately 10 classifiers. Rescoring 10 phone pairs corrects 78 out of 1,487 phone pairs, an improvement of 5.2%. This is comparable to the results achieved in the LVCSR tasks. However, with approximately 20% of the test data rescored, an improvement in accuracy of 1.1% (absolute) over the baseline ML system is observed. As shown in the graph, the benefits from rescoring with more than 10 augmented models are limited. This is believed to be due to over-training.[6]

Using an identical procedure to that used above for the ML baseline, an MMI baseline can also be rescored. Results are shown in figure 4. As before, rescoring the baseline with augmented models yields improvements in accuracy. The improvements, however, are much smaller

[5] If either of the posteriors, $P(\theta_t = \{j, m\}|\boldsymbol{O}; \boldsymbol{\lambda})$ or $P(\theta_\tau = \{k, n\}|\boldsymbol{O}; \boldsymbol{\lambda})$ are zero, then the joint posterior, $P(\theta_t = \{j, m\}, \theta_\tau = \{k, n\}|\boldsymbol{O}; \boldsymbol{\lambda})$ will also be zero. However, if both posteriors are one, the joint posterior will also be one and so the difference will be zero.

[6] Although large quantities of training data exist for the few first pairs, the number of training occurrences decreases rapidly for later phone pairs. It therefore becomes increasingly difficult to obtain robust classifiers that can reliably separate the phones.

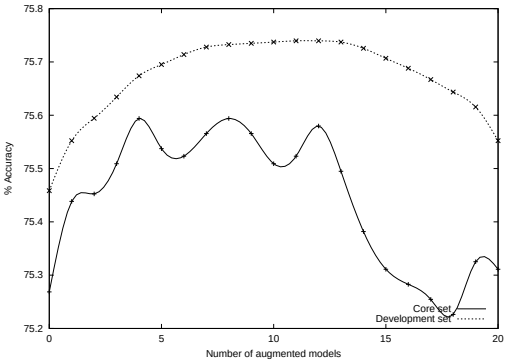



**Fig. 4** Rescoring of an MMI baseline for the TIMIT phone classification task

than in the ML case. This is because the MMI baseline performs much better than the ML baseline (24.7% versus 28.8%). There are therefore many fewer errors for the augmented models to correct. Rescoring using 8 augmented models yields an improvement of 0.3% (absolute) over the MMI baseline.

## 6 Conclusion

In this paper continuous rational kernels, an extension to rational kernels, are proposed. These allow task-dependent kernels for variable-length sequences of continuous observations to be defined. It is shown that, using the continuous rational kernel framework, $n$-gram and gappy-$n$-gram transducers map observation sequences into feature-spaces of state posteriors. With appropriate weighting, these posteriors allows derivatives of base acoustic models, with respect to the component priors and transition probabilities, to be calculated using only standard transducer operations. Derivatives with respect to the means and covariances can be calculated using a vector semiring. Combining these separate derivatives allows powerful generative score-spaces and kernels to be defined and calculated. Calculation using the continuous rational kernel framework has a significant advantage over direct calculation of score-space elements since it generalises well to higher dimensions—the same algorithms are used with different transducers. Initial experiments using SVM training and generative score-spaces on a large vocabulary speech recognition task indicate that the score-space and kernels detailed in this paper may be useful for speech recognition.

## A Fisher Derivatives

Consider an HMM as an example base acoustic model,

$$p(\boldsymbol{O};\boldsymbol{\lambda}) = \sum_{\boldsymbol{\theta}\in\boldsymbol{\Theta}} \prod_{t=1}^{T} a_{\theta_{t-1}\theta_t} p(\boldsymbol{o}_t|\theta_t;\boldsymbol{\lambda}) \tag{25}$$

where $\boldsymbol{\theta}$ denotes the latent state sequence associated with an observation $\boldsymbol{O}$, selected from the set of all possible state sequences, $\boldsymbol{\Theta}$. Introducing a Lagrange multiplier to enforce the sum-to-one constraint on the component priors, $\sum_{m=1}^{M} c_{jm} = 1$, and considering the derivatives of the log-likelihood of the acoustic model with respect to the component prior, $c_{jm}$, and the Lagrange multiplier, $\alpha$, yields,

$$\frac{\partial \ln p(\boldsymbol{O};\boldsymbol{\lambda})}{\partial c_{jm}} = \frac{\sum_{t=1}^{T} P(\{j,m\}|\boldsymbol{o}_t;\boldsymbol{\lambda})}{c_{jm}} + \alpha \tag{26a}$$

$$\frac{\partial \ln p(\boldsymbol{O};\boldsymbol{\lambda})}{\partial \alpha} = \sum_{k=1}^{M} c_{jk} - 1 \tag{26b}$$

Next, equating equations (26a) and (26b) to zero and rearranging yields,

$$-\sum_{t=1}^{T}\sum_{m=1}^{M} P(\{j,m\}|\boldsymbol{o}_t;\boldsymbol{\lambda}) = \alpha \sum_{m=1}^{M} c_{jm} = \alpha \tag{27}$$

Therefore the final, constrained, derivative is given by,

$$\nabla_{c_{jm}} \ln p(\boldsymbol{O};\boldsymbol{\lambda}) = -\sum_{t=1}^{T} P(\theta_t = s_j|\boldsymbol{o}_t;\boldsymbol{\lambda}) + \frac{1}{c_{jm}} \sum_{t=1}^{T} P(\theta_t = \{j,m\}|\boldsymbol{o}_t;\boldsymbol{\lambda}) \tag{28}$$

Similarly, derivatives with respect to the transition probabilities are given by,

$$\nabla_{a_{ij}} \ln p(\boldsymbol{O};\boldsymbol{\lambda}) = -\sum_{t=1}^{T} P(\theta_t = s_j|\boldsymbol{o}_t;\boldsymbol{\lambda}) + \frac{1}{a_{ij}} \sum_{t=1}^{T-1} P(\theta_t = s_i, \theta_{t+1} = s_j|\boldsymbol{o}_t;\boldsymbol{\lambda}) \tag{29}$$

## References

1. D. Povey, *Discriminative Training for Large Vocabulary Speech Recognition*, Ph.D. thesis, University of Cambridge, July 2004.
2. V.N. Vapnik, *Statistical Learning Theory*, John Wiley & Sons, 1998.
3. H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins, "Text classification using string kernels," *Journal of Machine Learning Research*, vol. 2, pp. 419–444, 2002.
4. K. Tsuda, T. Kin, and K. Asai, "Marginalized kernels for biological sequences," *Bioinformatics*, vol. 18, pp. S268–S275, 2002.
5. T. Jaakkola and D. Hausser, "Exploiting generative models in disciminative classifiers," in *Advances in Neural Information Processing Systems 11*, S.A. Solla and D.A. Cohn, Eds. 1999, pp. 487–493, MIT Press.
6. N. Smith and M. Gales, "Speech recognition using SVMs," in *Advances in Neural Information Processing Systems 14*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. 2002, pp. 1197–1204, MIT Press.

7. C. Cortes, P. Haffner, and M. Mohri, "Positive definite rational kernels," in *16th Annual Conference on Computational Learning Theory (COLT 2003)*, Washington D.C., August 2003, pp. 656–670.
8. C. Cortes, P. Haffner, and M. Mohri, "Rational kernels: Theory and algorithms," *Journal of Machine Learning Research*, vol. 5, pp. 1035–1062, 2004.
9. M. Mohri, F. Pereira, and M. Riley, "Weighted finite-state transducers in speech recognition," *Computer Speech & Language*, vol. 16, pp. 69–88, January 2002.
10. F.C.N. Pereira and M.D. Riley, "Speech recognition by composition of weighted finite automata," in *Finite-State Devices for Natural Language Processing*, E. Roche and Y. Schabes, Eds. MIT Press, 1997.
11. J.S. Garofolo *et al.*, *DARPA TIMIT Acoustic-Phonetic Continuous Speech Corpus CD-ROM*, 1993.
12. N.D. Smith and M.J.F. Gales, "Using SVMs to classify variable length speech patterns," Tech. Rep. CUED/F-INFENG/TR.412, Department of Engineering, University of Cambridge, April 2002.
13. V. Venkataramani, S. Chakrabartty, and W. Byrne, "Support vector machines for segmental minimum Bayes risk decoding of continuous speech," in *ASRU 2003*, 2003, pp. 13–18.
14. M. Mohri, "Finite-state transducers in language and speech processing," *Computational Linguistics*, vol. 23, no. 2, pp. 269–311, 1997.
15. M. Mohri, "Semiring frameworks and algorithms for shortest-distance problems," *Journal of Automata, Languages and Combinatorics*, vol. 7, pp. 321–350, 2002.
16. J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, 2004.
17. L.E. Baum and J.A. Eagon, "An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology," *Bull. Amer. Math. Soc.*, vol. 73, pp. 360–363, 1967.
18. L.R. Bahl, P. Brown, P. de Souza, and R. Mercer, "Maximum mutual information estimation of hidden Markov model parameters for speech recognition," in *Proc. ICASSP*, Tokyo, 1986.
19. O. Cappé, E. Moulines, and T. Rydén, *Inference in Hidden Markov Models*, Springer, 2005, Springer Series in Statistics.
20. G. Evermann, H.Y. Chan, M.J.F. Gales, B. Jia, D. Mrva, P.C. Woodland, and K. Yu, "Training LVCSR systems on thousands of hours of data," in *Proc. ICASSP*, 2005, pp. 209–212.
21. L. Mangu, E. Brill, and A. Stolcke, "Finding consensus among words: Lattice-based word error minimization," in *Proc. Eurospeech*, 1999, pp. 495–498.
22. N.D. Smith, *Using Augmented Statistical Models and Score Spaces for Classification*, Ph.D. thesis, University of Cambridge, September 2003.
23. A. Gunawardana, M. Mahajan, A. Acero, and J.C. Platt, "Hidden conditional random fields for phone classification," in *Interspeech*, 2005.
24. A.K. Halberstadt and J.R. Glass, "Heterogeneous acoustic measurements for phonetic classification," in *Eurospeech*, 1997, pp. 401–404.