Project acronym	SIMBAD
Project full title	Beyond Features: Similarity-Based Pattern Analysis and Recognition
Deliverable Responsible	Dipartimento di Informatica Università degli studi di Verona Strada le Grazie, 15 – 37134 Verona (Italy) http://www.di.univr.it/
Project web site	http://simbad-fp7.eu
EC project officer	Teresa De Martino
Document title	Generative Kernels
Deliverable n.	D2.2
Document type	Report
Dissemination level	Public
Contractual date of delivery	M 18
Project reference number	213250
Status & version	Definitive version
Work package, Deliverable responsible	WP2.1, UNIVR
Author(s)	M. Bicego, U. Castellani, M. Cristani, V. Murino, A. Perina
Additional contributor(s)	

Contents

1 Outline

This document results from work carried out in the context of the Task 2.1 (named "Generative Kernels") of the Work Package 2 (named "Deriving Similarities for Non-vectorial Data") of the SIMBAD project. A generative kernel is a kernel which measures the similarity between two objects via a trained generative model. These kernels are well suited to deal with non vectorial data, like strings, sequences, graphs, due to the intrinsic capabilities of generative models to properly characterize such kind of patterns. A generative kernel is defined through a two-steps process. In the first step, a set of generative models is trained; in the second one, the kernel is defined. This allows the definition of a hybrid generative-discriminative classification framework which has already shown their effectiveness on several applicative scenarios. Here we present contributions and results relative to both aspects of the generative kernel definition. The remainder of the report is organised as follows. Section 2 introduces the generative kernels; the contributions made in the context of the generative and the discriminative steps are presented in Sections 3 and 4, respectively. A final section with ongoing and future work will conclude the document.

2 Introduction

Kernel-based methods [61, 62, 53] have emerged in the last decade as a flexible and effective approach for addressing pattern recognition problems. These methods have been largely employed in supervised learning contexts (e.g., support vector classification and regression [61, 62, 53]) and recently also in the unsupervised learning context [4, 38, 16, 8].

A kernel function represents an inner product in a potentially infinite dimensional Hilbert space $\mathcal{H}$, to which the objects x_i of the problem are mapped through the mapping function Φ. In such Hilbert space the task may be more easily solved, since we can deal with nonlinearities. Many kernels have been defined in the last years, with different characteristics in terms of accuracy, theoretical properties and computational requirements. Among others, generative kernels [29, 31, 41, 20, 59] have recently emerged as a method to mix generative methods (like Hidden Markov Models or Bayesian Networks) and discriminative methods (like SVM), in order to merge the description capabilities of the former class of approaches with the discriminative skills of the latter class. Actually, generative models encode intuitively data regularities and a-priori knowledge through a graph structure, establishing a "correspondence" between parts of the model and features in the considered object. They deal successfully with problems such as missing, unlabeled and varying-length data, creating parametric class-conditional models. On the other side, discriminative approaches maximize the data separability, defining the separation bounds among classes, resulting in classification performance superior to the generative ones, in the case of enough training data [44].

Generative kernels represent a structured and well founded tool to merge these two classes of approaches, and are particularly well suited for handling structured data, such as sequences, strings, graphs, as the generative models employed in the kernel definition are able to capture and explicitly model their underlying structure. The basic idea is to assume a statistical model of the data (e.g., HMM, GMM etc.), and build a kernel on the probability distribution (or measure). Depending on which (dis)similarity measure between distributions is used, one obtains the Fisher kernel [29], the Probability Product Kernel and the Bhattacharyya affinity kernel [31], the Kullback-Leibler divergence kernel [41], the Jensen-Shannon divergence kernel [20], and the Marginalized kernel [59], [35], among others.

Other approaches implement the fusion of generative and discriminative principles as weighted convex combinations of generative and discriminative log likelihoods functions [63], minimizing the predictive conditional distribution when learning generative models [64] or considering different definitions of discriminative learning [65].

The general methodological approach to define a generative kernel consists of two steps:

1. *Generative Step*: in this step a generative model (or a set of generative models) is trained. The idea is to obtain a probabilistic generative model explaining and describing the objects of the problem.

2. *Discriminative Step*: in this step, the trained generative model is employed to compute and define a kernel between objects, to be used with a kernel machine to solve the task. In this context, the research lines are typically two: the first concerns the so called generative embeddings (also called generative Score Spaces [56]), where the main issue is to define how to project objects in a fixed length vector space – where inner product (or similar vector-based similarities) are used to compute the kernel. In the second

line, this embedding is implicit, and a direct kernel formulation is given (e.g., the Marginalized Kernel [59]).

Let's exemplify the process with the first introduced (and most famous) generative kernel, the Fisher Kernel [29]. This kernel measures the relation between objects by comparing them in the tangent space induced by the trained generative model. More in detail, the generative step is carried out by training a generative model for the whole training set. In the discriminative part, each object is represented by a feature vector, whose components are called Fisher scores. These scores are defined by derivatives of the log-likelihood of the generative model with respect to all individual parameters. The resulting kernel is then defined in such a feature space; the inner product was used in [29].

In the following sections, the contributions made in this context within the SIMBAD project are described. More in details, in section 3 the generative step is analyzed by exploiting different approaches for the training of the generative models. In section 4 the discriminative step is introduced by proposing two methods: i) a component based generative embedded for Hidden Markov Models, and ii) a new score space based on the free energy associated to the generative model. Finally in section 5 ongoing work and future issues have been envisaged.

3 Generative step

This section summarizes some results obtained in the generative step: more details may be found in the Simbad paper [7], attached to the report.

As a first consideration, it is worthwhile to note that, in the contest of generative kernels, most of the research efforts have been devoted to the discriminative step, namely to the definition of a proper generative embedding (the Score Space) or of a kernel. Nevertheless, improving the generative step may also lead to the improvement of the performances of the whole approach.

Let us briefly summarize how the generative step may be performed: we will focus on the well-known Fisher Kernel case [29]. In the original scenario defined by Jaakkola and Haussler in [29], the Fisher Kernel was computed on the basis of the log-likelihood of a single generative model, representing both competing classes (binary problem). This is the standard situation, also employed in most part of the generative kernels design methods. Another early version, by Fine et al. [23], defined the kernel on the basis of a generative model trained on a single class (the positive class). However, if more than one generative model is established, each representing a single class, more discriminative information may be extracted. Smith and Gales [54] exploited such idea in the binary classification case by proposing to employ in the Fisher Kernel the derivatives of the log-ratio of the two likelihoods calculated from the two competing models. It has been shown that this scheme may enhance the discriminative power of the Fisher Kernel. Other generalizations to multi-class models have been proposed, for example in [19], where multiple per class generative models were trained and used to derive Fisher kernel. In the paper, the authors also proposed a method to reduce the number of needed models (by randomly selecting a subset of per class models), in order to deal with the increased computational burden.

3.1 Clustering-based Construction of generative model

In this section, we describe a further contribution to the aforementioned scenario. The idea is to allow a generative framework to discover freely the natural structures or groups in the training set. This is achieved with a preliminary step of clustering, during which a large number of small hidden natural groups is extracted from the data, disregarding class label information. Subsequently, a single and simple generative model is trained for each group (as the groups tend to be small). The underlying intuition is simple: generative models are not used to discriminate between classes (this is left to the discriminative methods), but are used to finely describe the local structure of the data as an ensemble of clusters. In this way, the problem space is partitioned into small regions, each one characterized by a simple but well trained generative model.

Even if the proposed methodology may be general (and applicable to any generative kernel), we explore this direction focusing on the HMM-based Fisher Kernel case, showing promising and comparative results obtained from some experiments.

3.1.1 The methodology

The construction of our HMM-based generative kernel is realized in three steps: (1) discovering the data groups, (2) training a single HMM for each group, and (3) calculating and exploiting the related generative scores in the kernel definition. The three phases are reviewed in detail in the following.

The first step is to discover natural groups in the data by performing clustering of sequences. This step represents the most crucial part of the proposed methodology: it seems very reasonable to employ a process able to explicitly consider the generative model employed in the Fisher Kernel. In this sense, the clustering methodology employed here is based on HMM. The simplest and most widely used class of approaches for HMM-based clustering is the proximity-based clustering, where the main effort lies in devising good similarity or distance measures between sequences. With such measures, any standard distance-based method (as agglomerative clustering) can be applied. In this context, HMMs are employed to compute similarities between sequences, using different approaches (see for example [47, 2]), and standard pairwise distance-based approaches (as agglomerative hierarchical) are then used for identifying the final data clusters. The method employed in our study belongs to this class of approaches: first, we train one model for each sequence; the distance between a pair of sequences is then computed by evaluating the probability that the model trained on the first sequence generates the second sequence (and the other way around); finally, given the similarity matrix, complete link agglomerative clustering has been employed – more details can be found in [57, 47]. Clearly, the choice of the best number of clusters represents a problem, even though different indices/strategies have been proposed (see for example [30]). In our experimental evaluation, we let it vary in a proper range and report the different results.

Once estimated the natural groups inside the data set, a single HMM is trained for each group. Particular attention has been paid to the initialization of the training procedure – all details can be found in the Simbad paper [7].

To compute the Fisher Kernel given a set of trained models, we adopt the scheme proposed in [19], where the idea was to concatenate the scores obtained from each model. Here,

we adopt the same strategy, with the only difference that the models are not built on the classes but on the extracted clusters – see the paper [7] for more details.

Given the kernel, the classification task may be solved by using standard SVMs.

3.1.2 Experimental evaluation

The proposed methodology has been tested using a 2D shape recognition problem. The idea, in this case, was to extract the contour of the shape, transforming it to a sequence that is modeled by an HMM.

In particular, we studied the Chicken database, a very nasty problem: the results published in [40] report a baseline leave-one-out accuracy of $\approx 66\%$ by using the 1-NN on the Levenshtein (non-cyclic) edit distance. In our experiments, two different sequence representations are used to model contours, chain codes and curvature angles. Discrete HMMs are used to model the former clas of symbol sequences. In the second case, we derive curvature sequences as in [9, 43], subsequently modeled by continuous Gaussian HMMs.

Four methodologies to build the Fisher Kernel have been tested and compared:

1. *One model for the whole data set.* This is the standard methodology, proposed by Jaakkola and Haussler in their original paper [29]. One single model is built using all the data present in the training set. At the end only one model is trained.

2. *One class-model.* This is a generalization of the method proposed in [23], where a single model was trained using the data of the positive class. Since in that paper only binary problems were addressed, here we extend it to deal with the multi class case. To do that, we just select one of the classes, build the model for that class, and use this model to compute the Fisher Kernel. At the end only one model is trained. Clearly, depending on the chosen class, results may vary. Here we tried all the possibilities (reported in the tables as "Method 2 (class k)", indicating that an HMM has been trained using only the examples of the class k).

3. *C models: one per class* (C number of classes). This is the scheme proposed in [19], where one model per class is built. As explained before, the resulting Fisher Kernel is then defined as the inner product in the space obtained as a Cartesian product of the spaces resulting from each model (namely concatenating all Fisher Scores of all models). At the end, C models are trained, where C is the number of classes.

4. *K models: one per cluster* (K number of clusters). This represents the proposed approach.

In all cases the Fisher Score space has been normalized before the training of the linear SVM: this is required in order to make the Fisher Kernel work (see for example [54]). Accuracies have been computed by using the Averaged Holdout: the data set has been split in two random partitions, one used for training and one for testing. The process is repeated 10 times and the results are averaged. The results for continuous and discrete HMMs are shown in Table 1.

Method	# models	Accuracy (Std error of the mean)
1	1	0.759 (0.003)
2 (class 1)	1	0.755 (0.004)
2 (class 2)	1	0.758 (0.002)
2 (class 3)	1	0.755 (0.004)
2 (class 4)	1	0.734 (0.004)
2 (class 5)	1	0.752 (0.002)
3	5	0.775 (0.004)
4	19	0.798 (0.002)

(a)

Method	# models	Accuracy (Std error of the mean)
1	1	0.706 (0.006)
2 (class 1)	1	0.629 (0.004)
2 (class 2)	1	0.697 (0.009)
2 (class 3)	1	0.725 (0.006)
2 (class 4)	1	0.695 (0.005)
2 (class 5)	1	0.662 (0.004)
3	5	0.815 (0.004)
4	18	0.858 (0.002)

(b)

Table 1: Averaged accuracies (and standard errors) for different methods – see above. In the Clustering-based Fisher Kernel case (method 4), only the best result among the different clusterings is shown (in the range of 2-25 clusters): (a) Continuous HMMs applied to the Chicken Database with curvature sequences; (b) Discrete HMMs applied to the Chicken Database with chain code sequences.

3.1.3 Discussion

As a general comment, it is evident from the tables that the clustering-based building of the HMM pool results in a positive increase in the accuracy of the SVM based on Fisher Kernel; this is more evident in the discrete case. The obtained results are remarkable, considering the difficulty of the data set (for a comparative analysis, see Table 3 in [7]).

We also want to emphasize again that normalization of the Fisher Score spaces is essential (in whatever version, concatenated or not). Without normalization, the classification performance deteriorates significantly. This confirms the intuition provided in [54].

Concerning the Fisher Kernels defined on a single model (methods 1 and 2), it is interesting to observe that it does not make a significant difference to train the HMM either on the whole data set or on a single class. These models have discriminative powers which are, apparently, different, as combining them appears to be useful. This may rise the following question: is it reasonable to train a single "general shape HMM", namely to train an HMM on a different data set of shapes (or on a large collection of many databases)? In this way, the proposed approach may be considered as a pure feature extraction process.

The fact that Fisher Kernels built with only one model perform always worse than when built with more models confirms the intuition of [19], and is even more evident when using the clustering approach. Clearly, the resulting space may be very high-dimensional when several models are used, and the question arises of how to manage such a space. Here, we solve this by using an SVM based on the Fisher Kernel defined for the underlying Fisher Score space. As one can seen from the definitions in section 3.3, the Fisher Kernel of the combined space is just the average of the Fisher Kernels of the individual spaces. The kernel matrices have, of course, the same size determined by the size of the training set, and they are not related to the actual dimensionality of the data. Reasonably, this aspect would have become drastically crucial when studying the presented approach from a "generative embedding" point of view, namely when employing other classifiers (more sensitive to the curse of dimensionality) in the vector space derived from the generative model. Another

important observation is that in all the experiments we made, the best number of states in the clustering-based approach was two, indicating very small models. What we are doing in such a case is to increase the number of models while reducing the size of each model. This may alleviate the dimensionality issue.

In Figure 1 we plot the performances of the proposed approach in the chain code experiment while varying the number of clusters. As the presented approach can be understood as based on averaging kernels that are different, but that all make sense in one way or another, the number of kernels (and thereby clusters) should be sufficiently large to cover all aspects of the class distributions. After that the performance may stabilize, or may deteriorate somewhat as cluster sizes will shrink, resulting in models that may be more specific and thereby less useful for the following discriminative step.

The interplay between model size and number of clusters needs further study. In our approach, both are unsupervised procedures and may be based on other data than those available in a training set. Together they determine the kernel. However, the final performance obtained in the discriminative step by the SVM depends on the relation between this kernel and the size of the training set. So, all three have to be studied together: model size, number of models (clusters), and size of the training set.

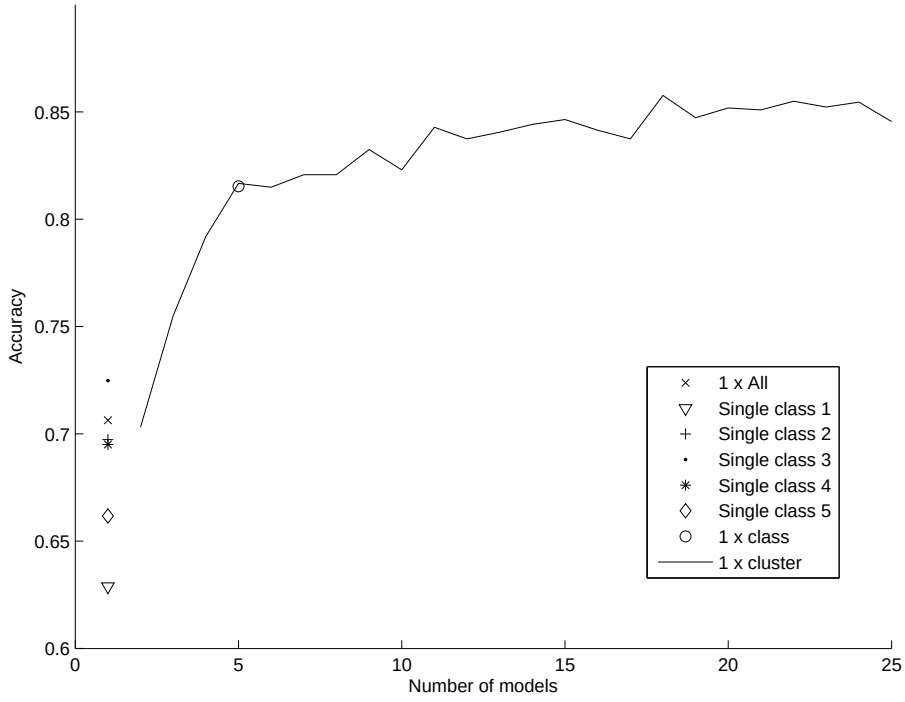

Figure 1: Performances on chain code experiment with varying number of clusters.

Looking at the whole procedure, there is a heavy data re-use: HMM training per sequence, clustering, HMM-training per group, parameter setting for SVM, and classifier training. Since we over-use the data a lot, we can benefit from weak and/or simple models, which is in fact confirmed in the experimental evaluation.

In summary, we should observe that the best results for clustering are obtained with small models (two state models) and a large number of clusters. The effect of such a result

is that the problem space is partitioned into small regions, each characterized by a simple but well trained generative model. Consequently, we can obtain an optimal description of the feature space (via a generative model), which is then partitioned by a discriminative method.

4 Discriminative step

In this section the results achieved in this context are summarized: all the details may be found in the Simbad papers [48, 50, 49, 12], attached to the report.

The discriminative step is aimed at employing the trained generative model to compute and define a kernel between objects, to be used with a kernel machine to solve the task. In this context, we mainly followed the research line of the so called generative embeddings (also called generative Score Spaces [56]), where the main issue is to define how to project objects in a fixed length vector space. Two research lines have been followed in this case: in the first, we proposed a set of generative embeddings for Hidden Markov Models, which exploit components of the model (like states or transitions) [12]. In the second, we defined a general score space based on the concept of Free Energy of a generative model [48, 50, 49].

4.1 Component based generative embeddings for Hidden Markov Models

Here we propose to derive a set of features from individual HMM class descriptions. In particolar, our proposal is to map each sequence into a HMM-induced vector space, whose features are derived from specific components of individual HMMs. Every feature measures the relevance of a specific component (like states or transitions) of a given HMM when an input sequence $\mathbf{O}$ is fed to this HMM, or, better, how a specific component contributes to the explanation of $\mathbf{O}$. A standard inner product results in a generative kernel, even if different classifiers may be used (actually, in [12] we tested many different discriminative classifiers in the obtained score space). We have shown that this method is robust and may drastically improve the standard HMM-based classification scheme. This is particularly evident when the original models are insufficient to solve the problem, usually due to small training sets, incorrect model topology or bad model assumptions. The described approach may be really useful in another crucial scenario, namely when different models – already trained on different data sets – are available, and new data (from a novel class) come. In this case, instead of training the models again (or training new models), we can just build the discriminant on the new data projected to the feature space through the already trained models. This has been illustrated in Section 4, where a feature space derived from a model of a *single* class is good enough to enable discrimination between different classes not modeled by HMMs. Such a scenario could be useful for using generic models in different cultural areas (e.g. for speech recognition), and also when a time adaption is required.

The approach proposed is an extension of some previous work [10, 11, 36], in which features were defined as functions measuring similarity between patterns and classes (or clusters). Here we go a step further and define features that describe relevance of particular components of the HMMs for explaining input sequences.

4.1.1 HMM-induced Vector Spaces

We will first start with some notation.

A discrete-time first order Hidden Markov Model [51] is a stochastic finite state machine defined over a set of K states $S = \{S_1, S_2, \cdots, S_K\}$. Let $\mathbf{Q} = (Q_1, Q_2, \ldots, Q_T)$ be a fixed state sequence of length T with the corresponding observations $\mathbf{O} = (O_1, O_2, \ldots, O_T)$. A HMM is described by a model $\boldsymbol{\lambda}$, determined by a triple $\{\mathbf{A}, \mathbf{B}, \boldsymbol{\pi}\}$ such that

- $\mathbf{A} = (a_{ij})$ is a matrix of transition probabilities;
- $\mathbf{B} = (b_j(o))$ consists of emission probabilities, in which $b_j(o) = P(O_t = o \,|\, Q_t = S_j)$ is the probability of emitting the symbol o when being in state S_j.
- $\boldsymbol{\pi} = (\pi_i)$ is the initial state probability distribution, *i.e.* $\pi_i = P(Q_1 = S_i)$.

Let $\mathcal{O} = \{\mathbf{O}_1, \ldots, \mathbf{O}_n\}$ be a set of observation sequences (of variable lengths) with the corresponding labels $\mathcal{Y} = \{y_1, \ldots, y_n\}$, indicating class memberships to one of C classes, $\omega_1, \ldots, \omega_C$. Let $\mathcal{O}^c = \{\mathbf{O}_1^c, \ldots, \mathbf{O}_{n_c}^c\}$ denote sequences of the class ω_c.

Here we assume that each class ω_c is described by a generative K-state Hidden Markov Model $\boldsymbol{\lambda}_c$, with the states $S^c = \{S_1^c, S_2^c, \cdots, S_K^c\}$, trained on the sequences from $\mathcal{O}^c$. Each $\boldsymbol{\lambda}_c$ is estimated via the Baum-Welch re-estimation algorithm such that $P(\mathbf{O}_i^c|\boldsymbol{\lambda}_c)$ is maximized for all $\mathbf{O}_i^c$ [51]. This results in a battery of C HMMs $\{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_C\}$.

We call "HMM-induced vector space" (`HMMVS`) the space defined by deriving a fixed-dimensional feature space from the trained generative HMMs. Every feature is extracted from a specific HMM and conveys information about the corresponding class.

`HMMVS` are based on "Component Information" features, $\mathcal{CI}$s, which describe some relevant information extracted from particular components of the models, in relation to the input sequence $\mathbf{O}$. A $\mathcal{CI}$ feature either characterizes some properties of the generation path of the sequence $\mathbf{O}$ through the model $\boldsymbol{\lambda}_c$ or the strength with which a specific component of $\boldsymbol{\lambda}_c$ "responds" to $\mathbf{O}$. More formally, $\mathcal{F}_{\mathcal{CI}}(\cdot, \boldsymbol{\lambda}_c) : \mathcal{O}^c \rightarrow \mathbb{R}^{m_c}$ is a model-dependent mapping defined by m_c components derived from $\boldsymbol{\lambda}_c$. The final HMM-induced vector space is a Cartesian product of all $\mathcal{CI}$-spaces (one for each class), $\mathcal{F}_{\mathcal{CI}}(\cdot, \boldsymbol{\lambda}_1) \times \ldots \times \mathcal{F}_{\mathcal{CI}}(\cdot, \boldsymbol{\lambda}_C)$ such that:

$$\mathcal{F}_{HMM}(\mathbf{O}) = \left[\mathcal{F}_{\mathcal{CI}}(\mathbf{O}, \boldsymbol{\lambda}_1),\, \mathcal{F}_{\mathcal{CI}}(\mathbf{O}, \boldsymbol{\lambda}_2), \ldots,\, \mathcal{F}_{\mathcal{CI}}(\mathbf{O}, \boldsymbol{\lambda}_C)\right]^T. \tag{1}$$

We introduce now four different $\mathcal{CI}$-mappings, leading to four different HMM-induced spaces (see [51] on how to compute the relevant quantities):

1. **`LL`-space** (Log-Likelihood space): This is our reference space, similar to the ones proposed in [10, 56, 11]. The $\mathcal{CI}$-feature becomes the logarithm of the probability that the model $\boldsymbol{\lambda}_c$ has generated the sequence $\mathbf{O}$, *i.e.* $\mathcal{CI}^{\mathtt{LL}}(\mathbf{O}, \boldsymbol{\lambda}_c) = \log P(\mathbf{O}|\boldsymbol{\lambda}_c)$. So, the extracted Component Information tells us how well a sequence is modeled by the given HMM. The resulting C-dimensional vector in `HMMVS` is defined as $\mathcal{F}_{HMM}^{\mathtt{LL}}(\mathbf{O}) = \left[\log P(\mathbf{O}|\boldsymbol{\lambda}_1), \ldots, \log P(\mathbf{O}|\boldsymbol{\lambda}_C)\right]^T$.

2. **`state`-space:** Here we measure the contribution of individual states to the most likely generation of the observation sequence. The $\mathcal{CI}$-feature describes how often (and with which probability) the model $\boldsymbol{\lambda}_c$ passes through a particular state S_i^c when observing

the sequence $\mathbf{O}$. We start by considering the variable γ_i^t, obtained from the forward-backward procedure. It represents the probability of being in state S_i^c at time t, given the sequence $\mathbf{O}$ of the length T and the model $\boldsymbol{\lambda}_c$. We have

$$\gamma_i^t(\mathbf{O}, \boldsymbol{\lambda}_c) = P(Q_t = S_i^c | \mathbf{O}, \boldsymbol{\lambda}_c). \tag{2}$$

The relevant information for the state S_i^c of a given model $\boldsymbol{\lambda}_c$ is therefore the sum over time of the probability that $\boldsymbol{\lambda}_c$ passes through that state while emitting $\mathbf{O}$, *i.e.*

$$\bar{\gamma}_i(\mathbf{O}, \boldsymbol{\lambda}_c) = \sum_{t=1}^{T} P(Q_t = S_i^c | \mathbf{O}, \boldsymbol{\lambda}_c) = \sum_{t=1}^{T} \gamma_i^t(\mathbf{O}, \boldsymbol{\lambda}_c). \tag{3}$$

Notice that $\bar{\gamma}_i(\mathbf{O}, \boldsymbol{\lambda}_c)$, the sum of γ_i^t over time t, can be interpreted as the expected number of transitions from S_i^c [51]. It is therefore a natural measure of the importance of state S_i^c in the process of deriving $P(\mathbf{O}|\boldsymbol{\lambda}_c)$. Hence, our $\mathcal{CI}$-feature is defined as $\bar{\gamma}_i(\mathbf{O}, \boldsymbol{\lambda}_c)$. If $\bar{\gamma}_i(\mathbf{O}, \boldsymbol{\lambda}_c)$ is scaled by T, then the $\mathcal{CI}$-feature becomes independent of sequence length. We, however, do not follow this idea here. It appears that sequence length can be very informative in the experiments, because it captures complexity of the structure encoded in a sequences. We are not disturbed by variable-length sequences as the resulting $\mathcal{CI}$-features can appropriately be scaled for the use of discriminative classifiers.

The total Component Information for the class ω_c is captured by a K-element vector:

$$\mathcal{F}^{\mathtt{s}}_{\mathcal{CI}}(\mathbf{O}, \boldsymbol{\lambda}_c) = \left[\bar{\gamma}_1(\mathbf{O}, \boldsymbol{\lambda}_c), \ldots, \bar{\gamma}_K(\mathbf{O}, \boldsymbol{\lambda}_c)\right]^T. \tag{4}$$

Finally, each sequence is mapped into a CK-dimensional feature vector which summarizes the importance of each state S_i^c for every model $\boldsymbol{\lambda}_c$ with respect to the input sequence $\mathbf{O}$, *i.e.* $\mathcal{F}^{\mathtt{s}}_{HMM}(\mathbf{O}) = \left[\mathcal{F}^{\mathtt{s}}_{\mathcal{CI}}(\mathbf{O}, \boldsymbol{\lambda}_1), \ldots, \mathcal{F}^{\mathtt{s}}_{\mathcal{CI}}(\mathbf{O}, \boldsymbol{\lambda}_C)\right]$.

3. `trans`**-space:** We focus now on the importance of individual transitions in a HMM. Hence, we characterize the property of $\boldsymbol{\lambda}_c$ by the frequency with which each connection is used in the process of generating $\mathbf{O}$. The basic variable, easily computed from the forward-backward algorithm, is $\xi^t_{(i,j)}$. It represents the probability of passing from state S_i^c at time t to state S_j^c at time $(t+1)$, given the observations and the model, *i.e.*

$$\xi^t_{(i,j)}(\mathbf{O}, \boldsymbol{\lambda}_c) = P(Q_t = S_i^c, Q_{t+1} = S_j^c | \mathbf{O}, \boldsymbol{\lambda}_c). \tag{5}$$

Similarly to the previous case we compute the sum over t, deriving:

$$\bar{\xi}_{(i,j)}(\mathbf{O}, \boldsymbol{\lambda}) = \sum_{t=1}^{T-1} P(Q_t = S_i^c, Q_{t+1} = S_j^c | \mathbf{O}, \boldsymbol{\lambda}_c) = \sum_{t=1}^{T-1} \xi^t_{(i,j)}(\mathbf{O}, \boldsymbol{\lambda}_c). \tag{6}$$

$\bar{\xi}_{(i,j)}(\mathbf{O}, \boldsymbol{\lambda}_c)$ can be interpreted as the expected number of transitions from state S_i^c to state S_j^c [51]. Therefore, our $\mathcal{CI}$-feature is $\bar{\xi}_{(i,j)}(\mathbf{O}, \boldsymbol{\lambda}_c)$. The total Component Information for the class ω_c becomes now a K^2-dimensional vector, defined as

$$\mathcal{F}^{\mathtt{T}}_{\mathcal{CI}}(\mathbf{O}, \boldsymbol{\lambda}_c) = \left[\bar{\xi}_{(1,1)}(\mathbf{O}, \boldsymbol{\lambda}_c), \ldots, \bar{\xi}_{(1,K)}(\mathbf{O}, \boldsymbol{\lambda}_c), \bar{\xi}_{(2,1)}(\mathbf{O}, \boldsymbol{\lambda}_c), \ldots, \bar{\xi}_{(K,K)}(\mathbf{O}, \boldsymbol{\lambda}_c)\right]^T. \tag{7}$$

The HMMVS is defined by all models, hence each sequence is mapped into a CK^2-dimensional feature vector, summarizing the importance of all transitions $\{S_i^c \to S_j^c\}$ for each model $\boldsymbol{\lambda}_c$ and a sequence $\mathbf{O}$, *i.e.* $\mathcal{F}^{\mathtt{T}}_{HMM}(\mathbf{O}) = \left[\mathcal{F}^{\mathtt{T}}_{\mathcal{CI}}(\mathbf{O}, \boldsymbol{\lambda}_1), \ldots, \mathcal{F}^{\mathtt{T}}_{\mathcal{CI}}(\mathbf{O}, \boldsymbol{\lambda}_C)\right]$.

4. `emit`**-space:** In some applications, emission probabilities may represent the most meaningful part of a HMM. Hence, we characterize the property of $\boldsymbol{\lambda}_c$ by the sum of emission probabilities at a given state. The $\mathcal{CI}$-feature is defined as $\tilde{\rho}_i(\mathbf{O}, \boldsymbol{\lambda}_c) = \sum_{t=1}^{T} b(O_t|S_i^c)$. The total Component Information for the class ω_c is a K-dimensional vector,

$$\mathcal{F}^{\mathtt{E}}_{\mathcal{CI}}(\mathbf{O}, \boldsymbol{\lambda}_c) = \left[\tilde{\rho}_1(\mathbf{O}, \boldsymbol{\lambda}_c), \ldots, \tilde{\rho}_K(\mathbf{O}, \boldsymbol{\lambda}_c)\right]^T. \tag{8}$$

The final HMMVS is described by all models, hence each sequence is mapped into a CK-dimensional feature vector, $\mathcal{F}^{\mathtt{E}}_{HMM}(\mathbf{O}) = \left[\mathcal{F}^{\mathtt{E}}_{\mathcal{CI}}(\mathbf{O}, \boldsymbol{\lambda}_1), \ldots, \mathcal{F}^{\mathtt{E}}_{\mathcal{CI}}(\mathbf{O}, \boldsymbol{\lambda}_C)\right]$.

The spaces presented above are the ones most intuitively derived from HMMs. Other spaces can be defined by following the same principle, e.g. such that Component Information is summarized over a couple of states or over a set of transitions entering a particular state. Another possible extension is a fusion of (parts of) the spaces. The flexibility in the definition of Component Information can be utilized to tailor this generic approach into a specific application.

Feature normalization in HMMVS can be important depending on the chosen classifier. Two usual techniques are standardization (zero mean and unit variance) and linear scaling to the $[0, 1]$ domain. Normalization is necessary for the LL-space, as it relies on unbounded logarithms of probabilities, which may become very large in modulus for small probabilities. `state-`, `emit-` and `trans`-spaces are bounded, because their features are summations over sequences of probabilities of a fixed maximal length. In this case, it is not clear whether normalization is required. In our experiments we compare normalized and non-normalized spaces, showing that the usefulness of this operation strictly depends on the chosen application and the classifier. Nevertheless, we remove non-informative features in HMMVS with zero (or close to zero in the machine precision) variances as judged on the training set. This may especially be the case e.g. for `trans`-space when particular transitions never occur.

4.1.2 Illustrative synthetic example

Now we illustrate both the advantages and drawbacks of the proposed approach with a synthetic example. Our goal is to highlight the situations for which classifiers in HMMVS are superior to the traditional MAP approach. If the training set is sufficiently large and class models are well estimated (*i.e.* with the correct number of states and topology, and well estimated probability functions), then the MAP approach is more beneficial.

We consider a two-class problem, in which sequences of length 10 are generated from the two 4-state Gaussian HMMs specified in Fig. 2. Since these two models are very similar (especially in the emission probability density models), the discrimination becomes difficult. Our experimental evaluation is specified as follows:

- Disjoint training and test sets are randomly generated. The training set varies from 2 to 500 sequences per class. The test set contains 1000 sequences, 500 examples per class.

$$A = \begin{array}{|c|c|c|c|}\hline 0.10 & 0.40 & 0.40 & 0.10 \\ \hline 0.20 & 0.40 & 0.40 & 0.00 \\ \hline 0.10 & 0.40 & 0.30 & 0.20 \\ \hline 0.50 & 0.30 & 0.10 & 0.10 \\ \hline \end{array} \quad \boldsymbol{\pi} = \begin{array}{|c|}\hline 0.25 \\ \hline 0.25 \\ \hline 0.25 \\ \hline 0.25 \\ \hline \end{array} \quad B = \begin{array}{|c|c|}\hline \mu_1 = 1 & \sigma_1^2 = 1 \\ \hline \mu_2 = 3 & \sigma_2^2 = 1 \\ \hline \mu_3 = 5 & \sigma_3^2 = 1 \\ \hline \mu_4 = 7 & \sigma_4^2 = 1 \\ \hline \end{array}$$

(a) Parameters of the HMM generating sequences of class 1.

$$A = \begin{array}{|c|c|c|c|}\hline 0.25 & 0.25 & 0.25 & 0.25 \\ \hline 0.25 & 0.25 & 0.25 & 0.25 \\ \hline 0.25 & 0.25 & 0.25 & 0.25 \\ \hline 0.25 & 0.25 & 0.25 & 0.25 \\ \hline \end{array} \quad \boldsymbol{\pi} = \begin{array}{|c|}\hline 0.25 \\ \hline 0.25 \\ \hline 0.25 \\ \hline 0.25 \\ \hline \end{array} \quad B = \begin{array}{|c|c|}\hline \mu_1 = 1.1 & \sigma_1^2 = 1 \\ \hline \mu_2 = 3.1 & \sigma_2^2 = 1 \\ \hline \mu_3 = 5.1 & \sigma_3^2 = 1 \\ \hline \mu_4 = 7.1 & \sigma_4^2 = 1 \\ \hline \end{array}$$

(b) Parameters of the HMM generating sequences of class 2.

Figure 2: Two-class synthetic data; sequences are generated by two HMMs. A is a probability transition matrix, $\boldsymbol{\pi}$ denotes the initial state probabilities and $(\boldsymbol{\mu}, \boldsymbol{\sigma})$ are the parameters of the Gaussian emission density function.

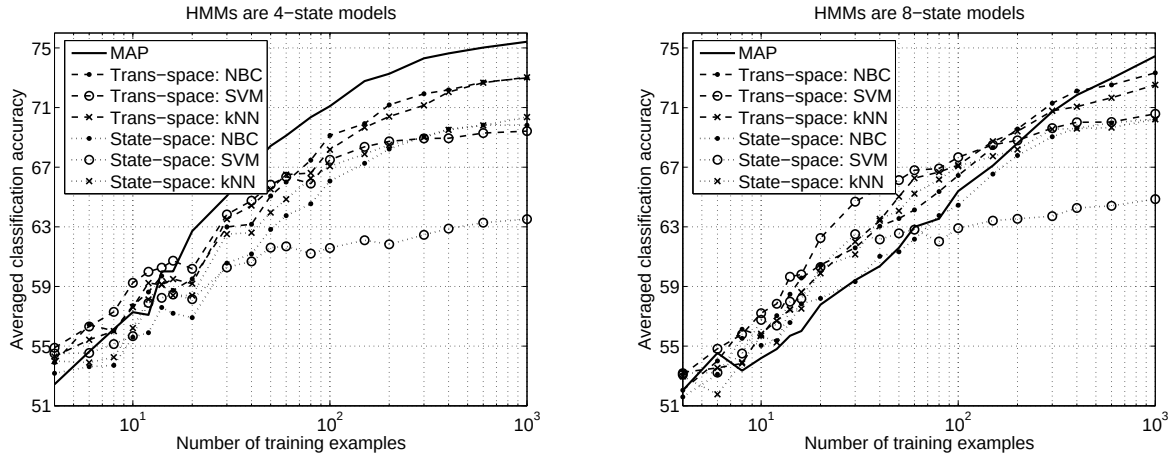

Figure 3: Averaged classification accuracy (in %) for the synthetic data as a function of the training size n_{tr}. Note that x-axis has a logarithmic scale. Concerning the correct 4-state class models (*left*), the standard errors of the average accuracy vary from 1.0-1.2% for $n_{tr} = 6$, via 0.9% (`trans`-space) or 1.2% (`state`-space) for $n_{tr} = 20$ to 0.55% for $n_{tr} = 100$ and, finally, 0.3-0.45% for large n_{tr}. Concerning the incorrect 8-state class models (*right*), the standard errors of the average results vary from 0.9-1.1% for $n_{tr} = 6$, via 0.7-0.8% for $n_{tr} = 20$ to $\approx 0.5\%$ for $n_{tr} = 100$ and, finally, 0.3-0.4% for large n_{tr}.

- Standard either 4-state or 8-state HMMs are trained per class, stopping at the likelihood convergence. The former case corresponds to a perfect model topology, while the latter simulates an incorrect model.
- Non-normalized `state`- and `trans`-spaces are constructed on the training set. The `state`-space is either 8D or 16D, while the `trans`-space is either 32D or 128D, depending on the number of states in the HMMs.
- Test sequences are classified either by the MAP approach applied to the two trained HMMs, or by classifiers trained in the HMM-induced vector spaces. The classifiers are chosen such that they do not suffer much from high-dimensional feature spaces

(zero-valued features that occur during training are neglected). The classifiers are:

1. the Naive Bayes Classifier (NBC – [27]), assuming that features are conditionally independent; individual class-conditional marginal densities are estimated by 1D histograms;
2. the Support Vector Machine (SVM – [52]) with a Gaussian kernel. ν is estimated by the leave-one-out nearest neighbor error on the training set, the scale σ of the Gaussian kernel is determined in a 20-step optimization based on the 5-fold cross-validation error estimation;
3. the k-Nearest Neighbor rule (k-NN – [24]). k is estimated on the training set by minimizing the leave-one-out classification error.

- The experiments are repeated 30 times and the average accuracy is reported.

Fig. 3 shows curves of average classification accuracy. We can observe that classification performance increases with the increasing training size, and that the `state`-space is inferior to `trans`-space, especially when SVM is trained. Most importantly, given the correct model topology ($K = 4$, left subplot), we can observe that the MAP results become increasingly better with a growing number of training sequences, finally outperforming the classifier in the `HMMVS`. For small sample size situations, however, the classifiers in the `trans`-space outperform the MAP approach. The right subplot of Fig. 3 shows the results for the situation where an incorrect model topology, 8-state HMMs, is used. In this case, the accuracy deteriorates for all classification strategies, but the MAP is suffering the most. The trained classifiers are capable of recovering from the improper estimates by using supervised training information in their discriminative learning. The k-NN and SVM trained in the `trans`-space provide a substantial improvement over the baseline MAP result for small and moderate training sizes and a similar result for large training sizes. This experiment supports the idea that our approach can be recommended for small training sets or when the chosen HMMs do not fit to the observed phenomena.

4.1.3 Experimental evaluation

The usefulness of `HMMVS` is evaluated on four applications: shape recognition (*Chicken* data), gesture classification (*Auslan* data), speaker verification (*Japan-Vowel* data) and EEG signal classification (*Alcoholic* data). These problems represent various scenarios, such as discrete symbol sequences versus continuous signals, difficult tasks versus easy tasks, small versus large number of classes, and finally, small or moderate training sets. The data sets are summarized in Table 2 and their further descriptions (including derivation of sequences) are given in Appendix A of the Simbad paper [12].

Experimental details and parameter setting

We assume that we deal with fully ergodic HMMs. Initialization is random both for the transition probabilities and initial state probabilities. In case of continuous signals, the emission probability models are initialized by a Gaussian Mixture clustering. In case of discrete symbol sequences, 20 independent training runs are performed, starting from a random

Table 2: Characteristics of the data. C denotes the number of classes. K is the number of states in the HMMs.

Data	C	size	# per class (min-max)	training/test	sequences	seq. length (min-max)	K
Chicken	5	446	61-117	20x 50%-50% holdout	discr. chain-codes	167- 248	3
Chicken	5	446	61-117	20x 50%-50% holdout	1D curvature	10-81	3
Auslan	10	270	27	20x 5-fold CV	22D sensors data	30-102	3
Japan-Vowel	9	640	61-118	fixed: 270/370	12D LPC coef.	7-29	5
Alcoholic	2	600	300	fixed: 600-600	2D EEG signals	256	9

initialization, picking the best likelihood model as the representative. Our implementation relies on the Murphy's Matlow Hidden Markov Model Toolbox[1].

For simplicity, the number of states is fixed for all classes in each problem. It is chosen beforehand by using some preliminary analysis or based on a priori knowledge (e.g. published papers).The same HMMs are used in the MAP scheme as in the proposed generative-discriminative framework.

Different strategies are used to estimate classification accuracy for each problem in order to make valid comparisons with the already published results. The idea is to perform a wide analysis of the discriminative power of the obtained feature spaces, not limiting the analysis to a specific kernel in the space. Therefore various classifiers, as implemented in `Prtools` [22], are tested in our HMM-induced vector spaces:

`lda` *Linear Discriminant Analysis* [24]. It assumes identical class conditional densities, modeled by Gaussians. The linear decision function is determined by the Bayes rule.

`loglc` *Logistic Linear Classifier* [27]. It models the log-odds (logarithm of the ratio of class posterior probabilities) as linear functions. The weights are optimized by ML.

`nbc` *Naive Bayes Classifier* [27]. It assumes that features are conditionally independent. Here the individual class-conditional marginal densities are estimated by 1D histograms.

`knn` *k-Nearest Neighbor rule* [24]. k is estimated on the training set by minimizing the leave-one-out classification error.

`svm` *Linear Support Vector Machine* [52]. This is the ν-SVM rule applied to a linear kernel (thus resulting in the most intuitive generative kernel). ν is estimated by the leave-one-out nearest neighbor error on the training set.

`rbsvm` *Radial Basis Support Vector Machine* [52]. This is the ν-SVM rule applied to a Gaussian kernel. ν is estimated by the leave-one-out nearest neighbor error on the training set. The scale σ of the Gaussian kernel is determined in a 20-step optimization based on the 5-fold cross-validation error estimation.

`1-svm` 1-*norm Support Vector Machine* [5]. This is a linear programming machine optimizing the ℓ_1-norm of the weights. It may serve as a feature selector.

[1]Downloadable from `http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html`

Table 3: Average classification accuracy (in %) in HMM-induced vector spaces for the *Chicken* data. The numbers in parenthesis describe the standard errors of the mean results. (N) denotes a normalised vector space.

Discrete sequences							
Classifier	`LL` (N) 5D	`State` (N) 15D	`State` 15D	`Trans` (N) 45D	`Trans` 45D	`Emit` (N) 15D	`Emit` 15D
`lda`	62.2 (1.0)	72.2 (0.8)	72.2 (0.8)	82.6 (0.6)	82.8 (0.6)	72.3 (0.7)	72.3 (0.7)
`loglc`	64.2 (0.9)	72.3 (0.8)	72.3 (0.8)	46.6 (4.7)	49.5 (5.0)	73.0 (0.5)	73.0 (0.5)
`nbc`	53.5 (0.7)	62.5 (0.7)	62.5 (0.7)	80.7 (0.6)	80.7 (0.6)	64.7 (0.5)	64.7 (0.5)
`svm`	61.9 (1.0)	68.7 (1.1)	64.2 (0.7)	80.4 (0.6)	69.4 (0.8)	69.0 (0.6)	62.6 (0.7)
`rbsvm`	62.0 (0.7)	72.8 (0.7)	71.3 (0.7)	81.1 (0.5)	74.9 (0.5)	75.8 (0.6)	74.0 (0.7)
`knn`	55.9 (0.7)	67.8 (0.7)	67.8 (0.9)	72.7 (0.9)	68.2 (0.7)	69.4 (0.7)	69.2 (0.5)
Continuous sequences							
Classifier	`LL` (N) 5D	`State` (N) 15D	`State` 15D	`Trans` (N) 45D	`Trans` 45D	`Emit` (N) 15D	`Emit` 15D
`lda`	70.7 (0.5)	72.5 (0.5)	72.5 (0.5)	75.5 (0.4)	76.1 (0.4)	73.0 (0.8)	73.0 (0.8)
`loglc`	69.5 (0.6)	71.9 (0.7)	71.9 (0.7)	36.5 (2.5)	36.0 (2.5)	71.9 (0.7)	71.9 (0.7)
`nbc`	58.0 (0.9)	70.0 (0.7)	70.0 (0.7)	71.5 (0.9)	71.5 (0.9)	68.9 (0.9)	68.9 (0.9)
`svm`	63.8 (0.8)	69.9 (0.8)	55.7 (0.9)	75.2 (0.6)	65.8 (0.8)	73.7 (0.6)	60.0 (0.9)
`rbsvm`	75.5 (0.5)	79.9 (0.6)	79.1 (0.5)	80.0 (0.6)	79.8 (0.4)	80.1 (0.5)	78.4 (0.5)
`knn`	71.0 (0.7)	77.4 (0.5)	77.3 (0.5)	74.4 (0.5)	77.0 (0.7)	77.2 (0.4)	75.2 (0.5)

Table 4: Average classification accuracy (in %) in HMM-induced vector spaces for the *Auslan* data. The numbers in parenthesis describe the standard errors of the mean results. (N) indicates a normalized vector space.

Classifier	`LL` (N) 10D	`State` (N) 30D	`State` 30D	`Trans` (N) 90D	`Trans` 90D	`Emit` (N) 30D	`Emit` 30D
`lda`	87.2 (0.4)	82.7 (0.6)	82.7 (0.6)	85.1 (0.4)	85.2 (0.5)	51.1(0.6)	10.0(0.0)
`loglc`	82.7 (0.5)	59.9 (2.0)	59.9 (2.0)	70.3 (1.1)	70.3 (1.1)	11.6(0.4)	11.6(0.4)
`nbc`	63.8 (0.5)	77.1 (0.6)	77.1 (0.6)	84.4 (0.5)	84.4 (0.5)	30.2(0.7)	10.0(0.0)
`svm`	84.5 (0.5)	78.5 (0.5)	52.2 (0.9)	82.2 (0.5)	51.3 (0.9)	60.6(0.5)	17.1(0.5)
`rbsvm`	86.1 (0.4)	82.1 (0.5)	77.1 (0.6)	84.5 (0.6)	77.3 (0.5)	62.0(0.6)	11.4(0.4)
`knn`	79.2 (0.5)	70.4 (0.9)	62.8 (1.0)	72.0 (0.7)	63.0 (1.0)	47.1(0.5)	11.5(0.4)

`lda`, `loglc` and `nbc` are not affected by a linear scaling of features, in contrast to classifiers that depend on a distance or inner product. Note also that both `lda` and `nbc` cannot be any longer considered as purely generative classifiers in our framework. The reason is that every feature in the `HMMVS` conveys information related to a specific class. If all models are used, features provide class-related responses of the trained HMMs. So, even if we model class-conditional densities, they incorporate information from all the classes thanks to the class-related features. The final classifier is trained in the `HMMVS`, after removing features with zero variances, which also reduces the space dimension.

Table 5: Average classification accuracy (in %) in HMM-induced vector spaces for the *Japan-Vowel* data. The training and test sets are fixed. (N) indicates a normalized vector space.

Classifier	LL (N) 9D	State (N) 45D	State 45D	Trans (N) 225D	Trans 225D	Emit (N) 45D	Emit 45D
lda	95.7	89.7	89.7	83.2	84.6	66.5	61.9
loglc	94.6	86.5	86.5	80.5	80.5	24.1	24.1
nbc	90.0	86.0	86.0	84.9	84.9	66.0	66.0
svm	96.8	90.0	68.1	90.0	64.3	83.2	65.7
rbsvm	96.2	91.1	92.2	89.5	91.6	87.6	88.1
knn	94.1	84.9	85.7	85.7	85.7	84.1	90.0

Table 6: Average classification accuracy (in %) in HMM-induced vector spaces for the *Alcoholic* data. The training and test sets are fixed. (N) indicates a normalized vector space.

Classifier	LL (N) 2D	State (N) 18D	State 18D	Trans (N) 162D	Trans 162D	Emit (N) 18D	Emit 18D
lda	58.8	59.0	59.0	59.8	60.0	58.2	58.2
loglc	57.0	59.5	59.5	46.2	46.2	60.0	60.0
nbc	54.7	61.8	61.8	61.5	61.5	60.8	60.8
svm	57.0	60.5	60.5	59.2	60.2	59.8	60.3
rbsvm	58.8	61.8	62.5	58.3	64.2	65.0	62.2
knn	58.3	62.3	63.2	61.5	59.2	62.2	58.5

Table 7: Comparison between the standard MAP results and the best results of the proposed scenario.

Problem	MAP	Best	Type	2nd Best	Type
Chicken (chain-code)	51.0 (0.8)	82.8 (0.6)	lda + trans-space	81.1 (0.5)	rbsvm + trans-space
Chicken (curvature)	57.4 (0.8)	80.1 (0.5)	rbsvm + emit-space	80.0 (0.6)	rbsvm + trans-space
Auslan	82.2 (0.4)	87.2 (0.4)	lda + LL-space	86.1 (0.4)	rbsvm + LL-space
Japan-Vowel	97.6	96.8	svm + LL-space	96.2	rbsvm + LL-space
Alcoholic	56.7	65.0	rbsvm + emit-space	64.2	rbsvm + trans-space

Results and discussion

Tables 3 – 6 report the results of our experimental evaluation. When appropriate (in case of cross-validation or hold-out experiments), averaged classification accuracies and their standard errors are displayed. The tables present classification performance in both normalized (by unit variance) and non-normalized state-, emit- and trans-spaces for the *Chicken*, *Auslan*, *Japan-Vowel* and *Alcoholic* data, correspondingly. The performance of the traditional MAP approach and the best performances in the proposed framework are shown in Table 7. While analyzing all results, the following observations can be made:

- **Discriminative classifiers in the HMMVS versus MAP.** In general, the proposed approach is better than the standard MAP classification scheme. The improvement

is negligible when HMMs are performing very well, as observed for the *Japan-Vowel* data. Improvements in accuracy are impressive when the estimated HMMs are weak, either undertrained or badly estimated, as it can be observed for the *Chicken* data.

- **Comparison between HMM-induced vector spaces.** Given C classes, each modeled by a K-state HMM, the dimensions of the `LL`-space, `state`-space, `emit`-space and `trans`-space are C, CK, CK and CK^2, respectively. If the number of classes is large and/or generative HMMs are well estimated, then the `LL`-space captures sufficient discriminative information in order to enable a very good classification performance. For instance, the best results for the 9-class *Japan-Vowel* data are reached in the `LL`-space. In this case, the high increase of the number of features in the `state`- and `trans`-spaces does not improve discrimination. On the other hand, when the HMMs are poor, then building a `HMMVS` via the model components works very well, as can clearly be observed for the *Chicken* and *Alcoholic* data. The best results are achieved by `rbsvm` in the `emit`-space and `trans`-space. Obviously, `trans`-space is the most flexible, but may also be inadvisable when the dimensionality grows exponentially (due to a high number of states and many classes). `emit`-space or `state`-space should be considered in such a case.
- **Normalization.** Normalization in the `LL`-space is recommended to prevent unbounded feature values. In other cases, normalization can be useful, depending on the task and the classifier. For instance, `svm`, the linear ν-SVM seems to especially benefit from feature scaling, as observed for the *Chicken*, *Auslan* and *Japan-Vowel* data.
- **Choice of classifier.** There is no single best classifier, although ν-SVM based either on linear (`svm`) or Gaussian kernel (`rbsvm`) often performs best or close to the best. In addition, also `lda` gives very good results.

4.1.4 Dealing with space dimensionality

A potential drawback of using HMM-induced vector spaces is the curse of dimensionality, especially when non kernel machines are employed. The log-likelihood space (`LL`-space) is C-dimensional, `state`-space and `emit`-space are CK-dimensional, while `trans`-space is CK^2-dimensional. The problem is even more enhanced when two or more feature spaces are fused via Cartesian product. This is relevant for density-based classifiers, such as the linear or quadratic discriminant, which suffer from a large number of features. In the `trans`-space, the training size has to increase exponentially in order to permit a reliable estimation of the class-conditional densities. Supervised and unsupervised feature reduction techniques are standard solutions to diminish the negative effect of high dimensionality [24]. Here, we will perform both unsupervised and supervised reduction, by using Principal Component Analysis (PCA) and Forward Feature Selection (FFS) based on the 1-NN classification criterion.

Other classifiers are less affected by the curse of dimensionality. The most notable example is SVM [62], whose complexity depends on the number of support vectors instead of the space dimension. It appears to perform well in high-dimensional spaces. A variant of SVM is a 1-norm SVM [5, 6], optimizing the ℓ_1-norm of its weight vector, which leads to a sparse solution. Hence, this method implements a feature selection mechanism inside the classifier training.

An alternative possibility of feature reduction is to focus on *a single HMM* to represent the information on sequences from *all the classes*. If a HMM is trained on a particular class such that it can capture the class characteristics well, then sequences from other classes may respond in a way that reflects the differences between the classes, hence provide discriminative information. Some substructures in a new sequence may be similar to the HMM-description of the chosen class, while other substructures may be very different. As a result, a few states or transitions may have very high Component Information, while others will have a low one. For a sufficiently rich HMM, several classes may respond fundamentally differently to this HMM, and may therefore be separable in the vector space induced by a single one-class HMM. This is in line with recent outcomes obtained for the Fisher Score approach presented in [19], where a subset of models have been successfully employed to build the space.

We used *Chicken* data to investigate the usefulness of the above-discussed feature reduction approaches for the construction of `HMMVS`. All the results and the comments may be found in Section 4 of [12]. A remarkable observation is the following: in summary, the experiments demonstrate that a Hidden Markov Model of a single class can be used to differentiate between classes in the HMM-induced space, provided that it is a sufficiently complex model. This holds even for the classes for which the model was never trained. This is possible because of different class-related responses to the model, as the extracted component features are informative enough to discriminate between the classes. This alleviates the demand of estimating a HMM for each class separately, which becomes hard to satisfy when many classes are present, or when new classes may appear. In these cases one or just a few generic HMMs can be used to represent all the objects from all the classes.

This opens the possibility of a very general class description, without using specific characteristics of the class at hand. In the case of 2D shape recognition, one complex HMM may be trained on a very large data set that contains most of the possible variations in curves that may be expected. This HMM is then hoped to characterize typical substructures in these data, that later can be used as features for a discriminative classifier trained on the HMM outputs.

4.1.5 Comparison with the Fisher Score approach

In this section we compare the proposed spaces with the Fisher Score spaces, as the latter represent the baseline approach, and their usefulness has been largely demonstrated in the HMM-based classification [29, 17, 18].

In order to have a thorough comparison, we repeat the experiments described in Section 4.1.3 by using a *normalized* version of the Fisher Score space. This is achieved by a normalization step applied to the space defined in [29], as proposed in [54]. In the original work, the typical application scenario is a two-class case: positive class versus negative class. A single generative model is trained on the examples of the positive class (or on both classes) and used to define a Fisher Score space, in which a linear SVM is employed. Although proposals exist that effectively exploit the multi-class scenario, e.g. [25], here we use a simple extension, which permits us to have a direct and clear comparison between the information extracted by Fisher Scores and Component Information features in our approach. In particular, we train one HMM per class, mapping each object to a Fisher Score vector depending on all models. Different classifiers are trained in the resulting space.

For each experiment, the Fisher Score space is computed by using exactly the same

Table 8: Percentage improvements of the HMM-induced vector space over the Fisher Score space.

Classifier	*Chicken* (chain-code)	*Chicken* (curvature)	*Auslan*	*Japan-Vowel*	*Alcoholic*	Average over experiments
`lda`	+1.2	-0.4	+77.2	+87.3	+2.2	+33.5
`loglc`	+22.6	+5.5	-14.0	+11.6	+5.3	+6.2
`nbc`	+32.8	-1.0	+44.1	+81.6	-3.0	+30.9
`svm`	+1.1	-3.9	-11.3	+7.6	-0.8	-1.5
`rbsvm`	+0.1	-1.5	-10.7	+13.5	+4.2	+1.1
`knn`	+2.5	+0.7	-14.5	+6.8	+1.7	-0.5
Avr. over classifiers	+10.0	-0.1	+11.8	+34.7	+1.6	+11.6

HMMs as in our HMM-induced spaces. To have a direct comparison, Table 8 shows the improvements obtained by our best spaces with respect to the Fisher Score approach, in terms of percentage. A positive value means that the proposed approach is able to improve the accuracy with respect to the Fisher Score. The accuracies are shown for all experiments and all classifiers. Moreover, the averages among experiments and among classifiers are also noted.

We can see that our HMM-induced space is generally better than the Fisher Score space, depending on the classifier. The latter is very competitive when using Support Vector Machines (similarly as in [29]) as observed for the 10-class *Auslan* data. In other cases, the Fisher Score approach possibly suffers from the high dimensionality of the space, leading to very bad classifiers. This is especially evident for `lda`, which estimates singular covariance matrices. Actually, it should be noted that in the Fisher Score case, the features are explicitly linked to the model parameters (because they are derivatives with respect to model parameters), whereas in our approach they can also be derived from general components, such as states or emission functions, leading to more manageable space dimensions.

In addition, we also want to emphasize that normalization of the original Fisher Score spaces [29] is essential. Without normalization, the classification performance deteriorates significantly for `rbsvm` and `svm`, as well as for other classifiers. When we compared classification performance in both original Fisher Score spaces and HMM-induced spaces (not reported due to lack of space), the latter led to far better results. As such, normalization is not always necessary in `HMMVS`.

4.1.6 Summary and open issues

In summary, the proposed component-based discriminative classification for Hidden Markov Models can be recommended as a good alternative to the classical HMM-based classification, especially for imperfect models or small sample size problems. In particular, this scheme may be successfully exploited in the applications, where pre-trained models are already available (for example when new data arrive) or are partially known (due to partial prior knowledge on the problem). Another possible field is the set of applications where some of the models are appropriate and others are not, due to either partial or poor knowledge about the problem or due to partial/inequal observability of the phenomenon. In such a scenario it seems possible to enrich the discrimination by building a good discriminative space from only the accurate

models.

New interesting questions, of both immediate and further scope, can now be posed. These are:

- Is it a sound strategy to employ models trained on many different data and classes, such as a "general shape HMM", for example?
- How can we benefit by incorporating label information in the generative step?
- How big should the representation set (*i.e.* the set used to learn the HMMs and to build the corresponding vector space) be with respect to the set used for training the classifiers? Moreover, should these be disjoint?
- Which are the best components to be employed for a given problem?

4.2 Free energy score spaces

Here we propose a novel score space which exploits the free energy associated to a generative model. Free energy is a popular score function representing a lower bound on the negative log-likelihood of the visible variables. It is minimized in variational model training and normally ensures good model fitting. Here, we assume variational Expectation-Maximization (EM) as a standard learning tool, hence, the free energy permits to embed the uncertainty in the model parameters under the form of entropic terms. Such terms decompose in an entropy set and a cross-entropy set. The former encodes ambiguity within the model, thus considering the cases in which overfitting or local minima occur during the learning. The latter encodes errors in the model's fit to the data, distributing such discrepancies across several terms, each one focused on a particular factor of the generative joint distribution.

The entropy and cross-entropy sets are employed as features of the final score. The resulting score space shows to be highly informative for discriminative learning, allowing to achieve compelling comparative results in heterogeneous classification tasks, overwhelming the best classification accuracy on well-known databases. In particular, our approach was applied to face two bioinformatics problems (exons/introns classification, homology detection), and to deal with a typical computer vision issue (scene/object recognition), and results are compared with the best state-of-the-art outcomes present in the literature in the respective areas.

4.2.1 FES^2: Free Energy Score Space

We denote a generative model as the joint distribution $P(y, x) = P(\theta) \prod_{t=1}^{T} P(y^{(t)}, x^{(t)}|\theta)$ where $x = \{x^{(t)}\}_{t=1}^{T}$ is a set of i.i.d training observations; $y = \{y^{(1)}, \ldots, y^{(T)}, \theta\}$ is the set of hidden quantities, with $y^{(t)}$ a set of arbitrary hidden variables associated to the t-th observation, and θ is a set of parameters shared across the observations. We assume as standard learning method the variational Bayesian inference, which approximates the posterior on the hidden quantities $P(y|x)$, often intractable, by a simpler variational distribution $q(y) = \prod_{t=1}^{T} q(y^{(t)})$, that can be easily evaluated.

Free energy [32] is a score function whose minimization ensures a large similarity between the exact posterior and the variational distributions, also expressing a lower bound on the

negative log-likelihood of the data $-\ln P(x)$. Free energy is commonly arranged as:

$$\mathcal{F} = \mathbb{KL}(Q,P) - \ln P(x) = \sum_{[y]} q(y) \ln \frac{q(y)}{P(y|x)} - \sum_{[y]} q(y) \ln P(x) = \sum_{[y]} q(y) \ln \frac{q(y)}{P(y,x)} \quad (9)$$

where $\mathbb{KL}$ stands for the Kullback-Leibler (KL) divergence, and $\sum_{[y]}$ indicates the sum over all the possible values assumed by each of the hidden variables.
Minimization of $\mathcal{F}$ is usually achieved by variational inference using EM algorithm [32] which alternates between optimizing $\mathcal{F}$ with respect to q and θ respectively, while holding the other fixed.
To make the inference tractable, q can be constrained to belong to a simplified family of distributions $\mathcal{Q}$, and the free energy $\mathcal{F}_{\mathcal{Q}}$ is introduced to denote this relation. $\mathcal{F}_{\mathcal{Q}}$ represents a bound on the loglikelihood, i.e. $\mathcal{F}_{\mathcal{Q}} \leq -\ln P(x)$, and the equality does hold only if $\mathcal{Q}$ is expressive enough to capture the true posterior distribution. Examples of families are the fully-factorized mean field form [33] or the structured variational approximation [26], where some dependencies among the hidden variables are kept.

Once the parameters are estimated, $\theta \sim \hat{\theta}$, and assuming i.i.d. data, we can re-arrange equation 9 as

$$\mathcal{F}_{\mathcal{Q}} = -\ln P(\hat{\theta}) + \sum_t \left(\sum_{[y^{(t)}]} q(y^{(t)}|\hat{\theta}) \cdot \ln q(y^{(t)}|\hat{\theta}) - \sum_{[y^{(t)}]} q(y^{(t)}|\hat{\theta}) \cdot \ln P(y^{(t)}, x^{(t)}|\hat{\theta}) \right) \quad (10)$$

In the above equation, the first term inside the parentheses is an entropy term on $\{y^{(t)}\}$ which encodes the ambiguity within the model, and the second one is a cross-entropy term which approximates the divergence between the parameters' estimate $\hat{\theta}$ and the sample $x^{(t)}$ [14]. Generative classification implies to consider the free energy only with respect to a single test sample $x^{(t)}$; therefore, we define the free energy contribute $\mathcal{F}^t_{\mathcal{Q}}$ as a single summation term of Eq.10 plus the $-\ln P(\hat{\theta})$ term.
In addition, if q and P factorize, ambiguity and divergence can be highlighted locally as further summation terms, evidencing portions of the generative process more ambiguous, that probably the current model has not adequately encoded, or discriminant.
For example, if the generative model is described by a Bayesian network, its joint distribution can be written as $P(v^{(t)} = \prod_i P(v_i^{(t)}|\mathbf{PA}_i)$, where $v^{(t)}$ denotes the set of the variables (hidden or visible) and $\mathbf{PA}_i$ are the parents of $v_i^{(t)}$. We can compute the contributions for each variable of the model in the cross-entropy term:

$$\sum_{[v_1^{(t)}]} q(v_1^{(t)} \cup \mathbf{PA}_1|\hat{\theta}) \cdot \ln P(v_1^{(t)}|\mathbf{PA}_1, \hat{\theta}) + \cdots + \sum_{[v_N^{(t)}]} q(v_N^{(t)} \cup \mathbf{PA}_N|\hat{\theta}) \cdot \ln P(v_N^{(t)}|\mathbf{PA}_N, \hat{\theta}) \quad (11)$$

Moreover, for discrete hidden variables, each summation of Eq. 11 can be further exploded as a summation of factors over the (let's say, D) values that the related variable may assume, e.g., for $v_n^{(t)}$,

$$q(v_n^{(t)} = 1, \cup \mathbf{PA}_n|\hat{\theta}) \cdot \ln P(v_n^{(t)} = 1|\mathbf{PA}_n, \hat{\theta}) + \cdots + \quad q(v_n^{(t)} = D, \cup \mathbf{PA}_n|\hat{\theta}) \cdot \ln P(v_n^{(t)} = D|\mathbf{PA}_n, \hat{\theta}) \quad (12)$$

In a similar fashion, the entropy term can be further decomposed into a sum of terms as dictated by the factorization of the particular family $\mathcal{Q}$ selected.

As visible from equations 10-12, the free energy contribution of a sample, $\mathcal{F}^t_{\mathcal{Q}}$, is organized as a summation of terms which can be written as follows:

$$\mathcal{F}_{\mathcal{Q}} = -\ln P(\hat{\theta}) + \sum_t \mathcal{F}^t_{\mathcal{Q}} = -\ln P(\hat{\theta}) + \sum_t \sum_i f^t_{i,\hat{\theta}} \tag{13}$$

where all the free energy pieces $f^t_{i,\hat{\theta}}$ derive from the finest decomposition (Eq.12).
Such information can be encapsulated in a score space that we call *free energy score space* or simply FES^2. In the case of a binary classification problem, we can follow the notation of [54]: we indicate as $\mathcal{F}_{(\mathcal{Q},\hat{\theta})}(x^{(t)})$ the score argument on $x^{(t)}$ that calculates all the free energy contributions with respect to a) a particular model, indicated by its parameters estimation $\hat{\theta}$, and b) a particular choice of the family of the posterior distributions $\mathcal{Q}$. Then, we define the free energy score operator $\phi^{FES^2}_{\hat{\theta}}(x^{(t)})$ that applies on the score argument $\mathcal{F}_{(\mathcal{Q},\hat{\theta})}(x^{(t)})$, by mapping a test sample $x^{(t)}$ as follows

$$\phi^{FES^2}_{\hat{\theta}} : x^{(t)} \rightarrow \left[\mathcal{F}_{(\mathcal{Q},\hat{\theta}_1)}(x^{(t)}), \mathcal{F}_{(\mathcal{Q},\hat{\theta}_2)}(x^{(t)})\right] \quad where \;\; \mathcal{F}_{(\mathcal{Q},\hat{\theta}_c)} = [\ldots, f^t_{i,\hat{\theta}_c}, \ldots], c = 1, 2 \tag{14}$$

in which the $-\ln P(\hat{\theta}_i)$ terms have been removed as constants and not dependent by the particular sample.
In a purely generative classification, the two terms $\mathcal{F}_{(\mathcal{Q},\hat{\theta}_c)}$ are summed over, producing two values which are then compared.
In our case the aim is to exploit the discriminative properties of the generative process itself, by evaluating separately the model subparts driven by the unique factorization properties of the graphical model, and by the particular choice of $\mathcal{Q}$; keeping the free energy terms separated as components of a multivariate score space permits to exploit these local discriminative properties of the generative process with respect to a particular sample The intuition is in fact that samples belonging to the same class are likely to behave similarly under different model components. Such intuition is supported by the theoretical justifications discussed in the following section.

4.2.2 Theoretical analysis

According to the terminology introduced in [60], the proposed FES^2 allows us to obtain a *model-dependent feature extractor* as different generative models lead to different feature vectors [55].
This family of feature extractors $\phi_{\hat{\theta}} : \mathcal{X} \rightarrow \Re^d$ maps the input data $x \in \mathcal{X}$ in a space of fixed dimension derived from the so-called plug-in estimate λ, which is the generative model of parameters $\hat{\theta}$ from which the features are extracted.
In order to evaluate these feature extractors, we refer to the performance measures proposed in [60], where the authors proved that the TOP kernel performs better than its respective plug-in estimates.

Let x be the observations, and $y \in \{-1, +1\}$ be the class label. Let us assume that the parametric model of the joint probability $P(x, y|\theta)$ of parameter θ is known; also assume

that the model $P(x, y|\theta)$ is regular and contains the true distribution. Consequently, the true parameter θ^* is uniquely determined [24]. If we learn this generative model, we can obtain $\hat{\theta}$, an estimate of θ^* which is estimated by T training examples drawn i.i.d. from $P(x, y|\theta^*)$. As the Fisher and TOP kernels are commonly used in combination with linear classifiers such as linear SVMs, [60] proposes as reasonable performance measure the classification error of a linear classifier $w^T \cdot \phi_{\hat{\theta}}(x) + b$ in the feature space $\Re^d$, where $w \in \Re^d$ and $b \in \Re$. For a general analysis, we assume that w and b are chosen by an optimal learning algorithm. In this case, the classification error $R(\phi_{\hat{\theta}})$ turns out to be

$$R(\phi_{\hat{\theta}}) = \min_{w,b} E_{x,y}\Phi[-y(w^T \cdot \phi_{\hat{\theta}}(x) + b)] \tag{15}$$

where $\Phi[a]$ is an indicator function which is 1 when $a > 0$, and 0 otherwise, $E_{x,y}$ denotes the expectation with respect to the true distribution $P(x, y|\theta^*)$, and the apex T indicates the transpose operator.

In [29], it has been demonstrated that the Fisher kernel (FK) classifier can perform at least as well as its plug-in estimate if the parameters of a linear classifier are properly determined. Later, in [60], such property has been revised more concisely within this framework as:

$$R(\phi_{\hat{\theta}}^{FK}) \leq E_{x,t}\Phi[-y(P(y = +1|x, \hat{\theta}) - \frac{1}{2})] = R(\lambda) \tag{16}$$

where λ represents the generative model used as plug-in estimate.

The property expressed in equation 16 holds also for our method, where $\phi_{\hat{\theta}}(x^{(t)}) = \phi_{\hat{\theta}}^{FES^2}(x^{(t)})$. Given a generative model and a particular choice of $\mathcal{Q}$, we can compute the plug-in estimate error after the learning of the two generative models of parameters $\hat{\theta}_{+1}, \hat{\theta}_{-1}$, assigning the label $\hat{y}$ using the following rule:

$$\hat{y} = \min_y \{\mathcal{F}^t_{\mathcal{Q},\hat{\theta}_{+1}}, \mathcal{F}^t_{\mathcal{Q},\hat{\theta}_{-1}}\} = \Phi[\sum \mathcal{F}_{(\mathcal{Q},\hat{\theta}_{+1})}(x^{(t)}) - \sum \mathcal{F}_{(\mathcal{Q},\hat{\theta}_{-1})}(x^{(t)})] \tag{17}$$

where the summations indicates that the factorizations addressed in Eq.14 are summed over. We will refer to this decision rule as the *free energy test*, where the model with the lower free energy for the data point is chosen[2].

Lemma 4.1 *Let M be the length of the feature vector $\phi_{\hat{\theta}}^{FES^2}(x^{(t)})$, $\mathcal{Q}$ be the particular family for the posterior distribution and $R_{\mathcal{Q}}(\lambda)$ the error of the free energy test. A kernel classifier employing $\phi_{\hat{\theta}}^{FES^2}$ derived from a model that contains the label as a latent variable is, asymptotically, at least as good as the classification performance of the free energy test.*

$$R(\phi_{\hat{\theta}}^{FES^2}) \leq E_{x,t}\Phi[-y(P(y = +1|x, \hat{\theta}) - \frac{1}{2})] = R_{\mathcal{Q}}(\lambda)$$

Proof

$$\begin{aligned} R(\phi_{\hat{\theta}}^{FES^2}) &= \min_{w,b} E_{x,y}\Phi[-y(w^T \cdot \phi_{\hat{\theta}}^{FES^2}(x) + b)] \leq E_{x,y}\Phi[-y(w^T \cdot \phi_{\hat{\theta}}^{FES^2}(x) + b)] \ \forall \ w, b \\ &\leq E_{x,y}\Phi[-y(w_g^T \cdot \phi_{\hat{\theta}}^{FES^2}(x) + b_g)] \ \ if \ \ w_g = [\overbrace{+1, \cdots, +1}^{M \ \ times}, \overbrace{-1, \cdots, -1}^{M \ \ times}]^T, \ \ b_g = 0 \\ &= R_{\mathcal{Q}}(\lambda) \end{aligned} \tag{18}$$

[2]The extension to a multiclass classification is straightforward.

□

With Lemma 1, we showed that $R(\phi^{FES^2}_{\hat{\theta}}) \leq R_{\mathcal{Q}}(\lambda)$. When the family $\mathcal{Q}$ is expressive enough to capture the true posterior distribution, the free energy test is equivalent to Maximum Likelihood (ML) classification and $R_{\mathcal{Q}}(\lambda) = R(\lambda)$. In this case, the result of lemma 3.1 states that the theoretical properties of the Fisher and Top kernels [29, 60] hold for $\mathbf{FES}^2$ too. In the other cases, the computation of the likelihood is intractable, and the free energy test, with the corresponding $R_{\mathcal{Q}}(\lambda)$, is used instead of the ML classification.

4.2.3 Generalizations

An important problem that score space based methods face effectively is the classification of variable-length sequences which are mapped into feature vectors of equal size. As evinced from Eq.14, our mapping produces a number of terms which depends on the dimensionality of the $t-$th data sample. Let us discuss this point with an example. In hidden Markov models (HMM) [51], the visible variables are the observed sequences $x^{(t)}_1, \ldots, x^{(t)}_{K(t)}$, the hidden variables are the hidden state sequences $s^{(t)}_1, \ldots, s^{(t)}_{K(t)}$, and the parameters θ are the prior state distribution π, the state transition probability matrix $\mathbf{A} = a_{\{ij\}}$, and the emission probabilities $\mathbf{B} = b_{\{iv\}}$. The free energy of each sample x^t of an HMM can be written as follows

$$\begin{aligned}\mathcal{F}^t_{EX} &= \sum_{[s]} q(s^{(t)}_1) \ln q(s^{(t)}_1) + \sum_{[s]} \sum_{k=1}^{K(t)-1} q(s^{(t)}_k, s^{(t)}_{k+1}) \ln q(s^{(t)}_k, s^{(t)}_{k+1}) - \sum_{[s]} q(s^{(t)}_1) \ln \pi_{s^{(t)}_1} \\ &\quad - \sum_{[s]} \sum_{k=1}^{K(t)-1} q(s^{(t)}_k, s^{(t)}_{k+1}) \ln a_{s^{(t)}_k, s^{(t)}_{k+1}} - \sum_{[s]} \sum_{k=1}^{K(t)} q(s^{(t)}_k) \ln b_{s^{(t)}_k, x^{(t)}_k} \end{aligned} \tag{19}$$

In this case, we employed the exact posterior (EX) distribution so that free energy minimization is equivalent to the usual Baum-Welch training algorithm [51, 42] and $\mathcal{F}_{EX} = -\ln P(x)$.[3] Looking at Eq.19, it is evident how each sample would be mapped in a space whose dimension depend on the length of the sample $K(t)$. To solve this problem and to normalize the score vectors, before the application of the score operator, we simply perform the sums over k in the free energy, for each sequence.
We call this operation "wrapping" and in the following we will refer to the resulting score space as $\mathbf{wFES}^2$. More generally, the wrapping operation is applicable in each sum involved in $\mathcal{F}^t_{\mathcal{Q}}$, so varying the length M of the resulting feature vector. The longer the feature vector, the finer the level of detail with which the generative process for the $t-$th sample is represented.
In other words, one can consider as the first, coarsest, level of detail the one which provides the entropy and the cross-entropy terms (Eq.10), and, as second level of detail, the factorization induced by the topology of the generative model and by the choice of $\mathcal{Q}$ (Eq.11). Finally, one can consider the finest decomposition separating all the sums over the values of the hidden variables (Eq. 12).

[3] Further reading on variational learning for hidden Markov models can be found in [39].

The choice of the most convenient level of detail derives from intuitive considerations: the contributions of the several hidden variables may be summed up when their cardinality is high (e.g., sum over pixel variables in vision), and may lead to curse of dimensionality issues, or when each sample has a different length.

In any case, it is easy to show that the resulting feature vector $\phi_{\hat{\theta}}^{wFES^2}(x)$ must at least match the performance obtained by the purely generative approach.
The proof makes use of a rectangular block diagonal matrix, whose blocks are M ones-vectors e_n, where n represents the dimension. This matrix can be easily obtained as $\mathbb{I}_M \otimes e_n$, where $\otimes$ represents the Kronecker product, and $\mathbb{I}_M$ is the identity matrix of size M.

Lemma 4.2 *Let M be the length of the feature vector $\phi_{\hat{\theta}}^{wFES^2}(x^{(t)})$, $\mathcal{Q}$ be the particular family for the posterior distribution, and $R_{\mathcal{Q}}(\lambda)$ be the error of the free energy test. If $M \geq 2$[4], a kernel classifier employing $\phi_{\hat{\theta}}^{wFES^2}$ derived from a model that contains the label as a latent variable is, asymptotically, at least as good as the classification performance of the full free energy test.*

$$R(\phi_{\hat{\theta}}^{wFES^2}) \leq R_{\mathcal{Q}}(\lambda)$$

Proof

$$\begin{aligned} R(\phi_{\hat{\theta}}^{wFES^2}) &= \min_{\tilde{w},b} E_{x,y}\Phi[-y(\tilde{w}^T \cdot \overbrace{(\phi_{\hat{\theta}}^{wFES^2}(x)^T \cdot (\mathbb{I}_M \otimes e_n))^T}^{2M} + b)] \\ &\leq E_{x,y}\Phi[-y(\tilde{w}_g^T \cdot \overbrace{(\phi_{\hat{\theta}}^{wFES^2}(x)^T \cdot (\mathbb{I}_M \otimes e_n))^T}^{2M} + b_g] \;\; if \;\; \tilde{w}_g = [\overbrace{+1,\cdots,+1}^{M \;\; times}, \overbrace{-1,\cdots,-1}^{M \;\; times}]^T, \;\; b_g = 0 \\ &= R_{\mathcal{Q}}(\lambda) \end{aligned} \tag{20}$$

□

For hidden Markov models, n represents the variable length of a sequence $K(t)$, but in general it can represent each contribute of the $\mathcal{F}_{\mathcal{Q}}^t$ we want to sum up, as previously described.

Another generalization concerns the usage of a single generative model as descriptor for all the classes. In FES^2, this can be exploited in a straightforward way (similarly to what proposed in [29]), because the interest is in the difference of the generative process between a pair of examples $x^{(t_1)}$ and $x^{(t_2)}$, rather than in the difference of the posterior probabilities for the label, which is all that is required in a pure generative approach, i.e., using one model for each class.

4.2.4 Experiments

To evaluate our approach we focus on four standard datasets considering as comparative results the classification accuracies provided by the datasets' authors, those estimated with the plug-in estimate λ, and those obtained using the Fisher and TOP Kernel [29, 60] in their original definition (indicated with **FS** and **TK**, respectively).
Support vector machines (SVMs) with RBF kernel [52] are used as discriminative method

[4] If M=1 our approach reduces to a similarity-based approach [10]

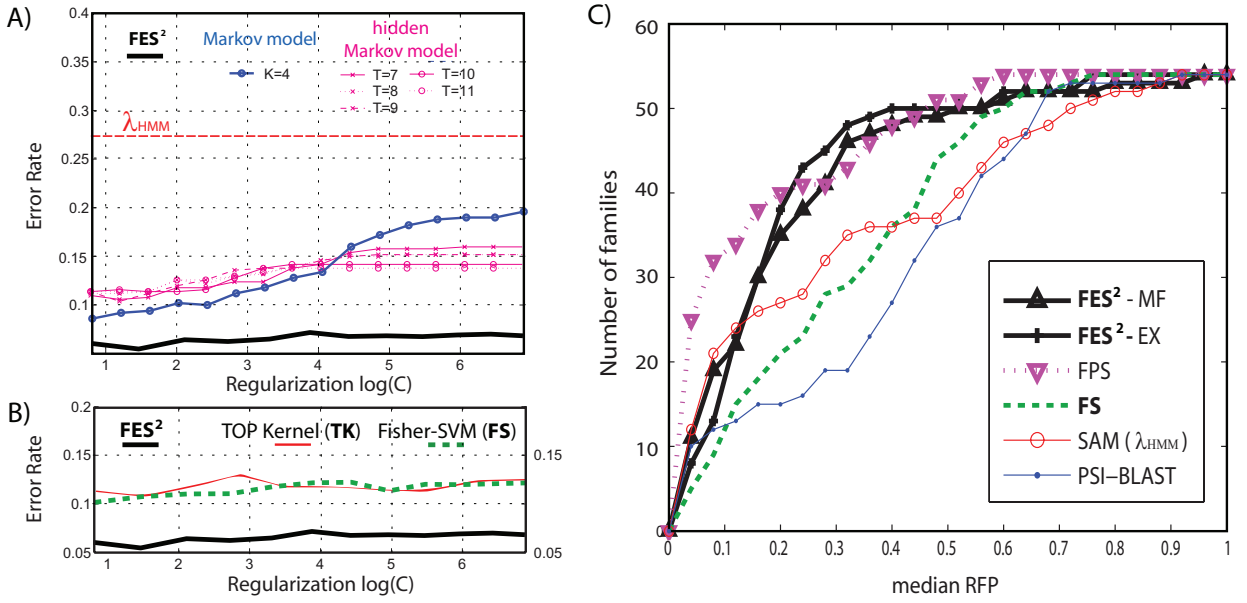

Figure 4: A) SVM error rates for (wrapped) FES^2 and Probability product kernels [31] using Markov models (we reported only their best result) and hidden Markov models as plug-in estimates. In this plot, T represents the parameters used in the proposed kernel [31], and K is the order of the Markov chain. For λ_{HMM} the x-axis does not apply. B) Performances of **FS** and **TK**. C) Relative performances of the five homology detection methods. Each profile plots the total number of families for which a given method exceeds a median RFP score (see text for more information).

on all the score spaces (**FES²**, **FS**), in order to gather flexibility when dealing with variable-length sequences [55]. We also used hidden Markov models (HMMs, Experiments 1-3) [51] or latent Dirichlet allocation (LDA, Experiment 4) [13] as plug-in estimate (λ_{HMM}, λ_{LDA}), and we employ the same validation procedure used in the original papers for comparison. For both **FS** and **FES²**, we learned a single generative model, as explained in section 4.2.3, since this leaded to the best results. The "correct" number of states for the HMM has been chosen considering a validation set extracted from the training set.

Experiment 1: E. coli promoter gene sequences. The first dataset is the E. coli promoter gene sequences (DNA) with associated imperfect domain theory [58]. The task is to recognize promoters in strings that represent nucleotides (A, G, T, or C). A promoter is a genetic region which initiates the first step in the expression of an adjacent gene (transcription). The input features are 57 sequential DNA nucleotides (fixed length). A special notation is used to simplify specifying locations in the DNA sequence. The biological literature counts locations relative to the site where transcription begins. Fifty nucleotides before and six following this location constitute an example. Results, obtained using leave-one-out (LOO) validation, are reported in Table 9 and confirm that **FES²** deals successfully with fixed size genetic sequences, outperforming with a large margin the other score spaces. In particular we experimentally showed the relation $R(\phi_{\hat{\theta}}^{wFES^2}) > R(\phi_{\hat{\theta}}^{FES^2})$ which intuitively holds since for **wFES²** less optimization factors $w's$ are involved (See Eq.18-20).

E.Coli	λ_{HMM}	**FES²**	**wFES²**	**FS**	**TK**
Accuracy	67,34%	94,33%	85,80%	79,20%	85,30%

Table 9: Promoter Region classification results. The wrapping operation has been performed over the sequence length (see Sec.4)

Experiment 2: Introns/Exons classification on HS³D data set. We considered the labeled gene sequences from the HS^3D data set[5] [31]. The variable sized arrays take symbols from a 4-letter alphabet (G,A,T,C). The goal here is to distinguish between gene sequences of two different types, i.e., introns and exons, using raw sequences of varying lengths (from dozens of characters to tens of thousands of characters). For the sake of comparison, we adopted the same experimental setting of [31]. In Fig.1-A (top right), we reported the results obtained in [31] (overall error rate, OER, 7,5%) and by the generative classification (λ_{HMM}, OER 27,59%) together with the results obtained by the proposed method (OER 6,12%). In Fig. 1-B (bottom right), we compared our method with **FS** (OER 10,06%) and **TK** (OER 12,82%).

Experiment 3: Homology detection on SCOP 1.53. We tested the ability of **FES²** to classify protein domains into superfamilies in the Structural Classification of Proteins (SCOP)[6] version 1.53. The sequences were selected using the Astral database, removing similar sequences using an E-value threshold of 10^{-25}. This procedure resulted in 4352 distinct sequences, grouped into families and superfamilies. For each family, the protein domains within the family are considered positive test examples, and the protein domains outside the family, but within the same superfamily, are taken as positive training examples. The data set yields 54 families containing at least 10 family members (positive test) and 5 superfamily members outside of the family (positive train) for a total of 54 One-Vs-All problems. The experimental setup is similar to that used in [28], except for one important difference: in the current experiments, the positive training sets do not include additional protein sequences extracted from a large, unlabeled database. Therefore, the recognition tasks performed here are more difficult than those in [28]. In order to measure the quality of the ranking, we used the median RFP score [28] which is the fraction of negative test sequences that scores as high as or better than the median-scoring positive sequences. We used SVM decision values as score.
Results confirms that **FES²** outperforms task-specific algorithms (PSI-Blast [1] and SAM [34]) and the Fisher score (**FS**,[28]) with statistical significance[7]. In particular, the poor performance of [28] can be explained with the under trained HMMs [15]; whereas **FES²** is robust to this problem. We repeated the test using two choices of $\mathcal{Q}$. The approximate mean field factorization and the exact posterior (indicated with **FES²**-MF and **FES²**-EX, respectively, in Fig.1-C) show how the specific choice does not affect too much the performances.

[5] `www.sci.unisannio.it/docenti/rampone`

[6] `http://scop.mrc-lmb.cam.ac.uk/scop/`

[7] *p-values* respectively of 5.1e-9, 8.3e-7, 1.1e-5, **FES²** and FPS [3] results are statistically equal.

Graz dataset	**FES²** - Z=15	**FES²** - Z=30	**FES²** - Z=45	[37]	[46]
Bikes	86,1% (1,8)	86,5% (2,0)	**89,1%** (2,3)	86,3% (2,5)	86,5%
People	83,1% (3,1)	82,9% (2,8)	**84,4%** (2,0)	82,3% (3,1)	80,8%
Scenes dataset	λ_{LDA}	**FES²**	**FS**	[45]	[37]
Natural	63,93%	**95,21**%	90,10%	89,00%	84,51%
Artificial	67,21%	**94,38**%	90,32%	89,00%	89,43%

Table 10: Object/scene recognition results (in square brackets the standard deviation is reported). Our approach is insensitive to the number of topics (Z). For scene recognition, results obtained with Z=40 are shown.

Experiment 4: Scene/object recognition. Our final set of experiments have been carried out using the Graz dataset[8], and also on the dataset proposed in [45]. For both tests, we used Latent Dirichlet allocation (LDA) [13] as generative model; details of the free energy formulation for LDA can be found in [13]. We extracted SIFT features from 16x16 pixel windows computed over a grid spaced of 8 pixels; we used 175 codewords ($W = 175$) varying the number of topics Z.
Graz dataset has two object classes, bikes (373 images) and persons (460 images), and a background class (270 images)[9]. The range of scales and poses at which exemplars are presented is very diverse, e.g., a "person" image may show a pedestrian at a certain distance, a side view of a complete body, or just a closeup of a head. For this database, we perform two-class detection (object vs. background) using an experimental setup consistent with [37, 46]. We generate ROC curves by thresholding raw SVM output, and report the ROC equal error rate averaged over ten runs. The results are shown in Table 10. Note that the standard deviation is quite high because the images in the database have different complexity, so the performance for any single run is dependent on the composition of the training set.
As final test, we performed scene recognition on the datasets proposed in [45], composed by two (Natural and Artificial scenes) 4-class datasets. The results are reported in Table 10 where for the first time we employed Fisher-LDA in a vision application. Once again, **FES²** outperforms Fisher score space and other state-of-the-art discriminative methods [45, 37].

More experiments Some more experiments on image related problems, with really satisfying results, may be also found in the Simbad paper [50]. Finally, a preliminary analysis on a simplified version of this score space, applied to HMM and Gaussians, has been presented in the simbad paper [48].

5 Ongoing work and future issues

Currently, we are addressing one further issue, related to normalization aspects of the generative embeddings. In the following subsections, we will present shortly the idea and some preliminary results, whereas the whole work will be presented in the final deliverable of WP2.

Morevoer, we are also starting to study how to boost the performances of generative kernels by combining in a proper manner the contributions of the kernels derived from

[8] `http://www.emt.tugraz.at/ pinz/data/GRAZ_02/`
[9] The car class is ignored as in [37]

different models. The idea is the following: we have shown in Section 3 that the use of multiple generative models (one for each class or one for each cluster) may improve the performances of the Fisher Kernel. In such case, the scores coming from each model are concatenated to obtain a single feature vector, and the kernel is the inner product in such concatenated space. Therefore, it may be also seen as the sum of a set of kernels, each one computed on scores derived from a single model. The interesting issue we are studying is the following: is it reasonable to weight differently the various kernels in such summation? And, most importantly, how can we compute the proper weights? One straightforward direction is to employ the techniques recently proposed in the kernel combining context [21]; an alternative direction may be to link the weights to some characteristics of the related generative model (like the training likelihood), in order to fully exploit the fact that we are dealing with generative kernels. This part is at the very beginning, and possible significant results will be provided in the final WP2 deliverable.

5.1 Space Normalization

Even if the success of the generative kernels has been proven on many applications there is still room for improvement. Actually, the different features of the derived generative embedding space could have different characteristics in terms of discriminative and descriptive power. Sometimes, these characteristics could not be completely highlighted by a simple inner product, and some space transformations may be useful. Actually, a space normalization step (centering and scaling) has been proposed in [54] that leads to an improved version of the Fisher Kernel. This has been also been confirmed by a set of experiments performed by us (an partially reported in our paper [12]), briefly described below. In particular we compared the `trans`-space proposed in Sect. 4.1 with the non normalized Fisher Score space in an experiment involving HMM.

5.1.1 `trans`-space vs Fisher Score space: the normalization issue

Again, in order to compare `trans`-space with the Fisher Score approach, we extend the binary case defined in [29] to the multi-class case by training one model for each class (and extracting the Fisher Scores for each model) and allowing for the use of different classifiers. We focus on a subset of all parameters, *i.e.* the transition probabilities $\{a_{ij}\}$. We therefore compare the discriminative power of the Fisher Score space and the `trans`-space: they have the same dimension and they rely on similar information extracted from the same entities.

Focusing on the state transition distribution ($\boldsymbol{\alpha} = \{a_{ij}\}$), the related Fisher Score becomes (see [17]):

$$\frac{\partial \log P(\mathbf{O}|\boldsymbol{\lambda})}{\partial a_{ij}} = \sum_{t=1}^{T-1} \xi^t_{(i,j)}(\mathbf{O}, \boldsymbol{\lambda}) \frac{1}{a_{ij}} = \frac{\bar{\xi}_{(ij)}(\mathbf{O}, \boldsymbol{\lambda})}{a_{ij}}. \tag{21}$$

Note that the above equation is very similar to (6) proposed in Sect. 4.1, except for the scaling by a_{ij}. In the terminology presented in Section 4.1, the corresponding K^2-dimensional Fisher Score is defined as

$$\mathcal{F}^{FScore}_{HMM}(\mathbf{O}) = [\mathcal{F}^{FScore}_{\mathcal{CI}}(\mathbf{O}, \boldsymbol{\lambda}_1), \ldots, \mathcal{F}^{FScore}_{\mathcal{CI}}(\mathbf{O}, \boldsymbol{\lambda}_c), \ldots \mathcal{F}^{FScore}_{\mathcal{CI}}(\mathbf{O}, \boldsymbol{\lambda}_K)]^T, \tag{22}$$

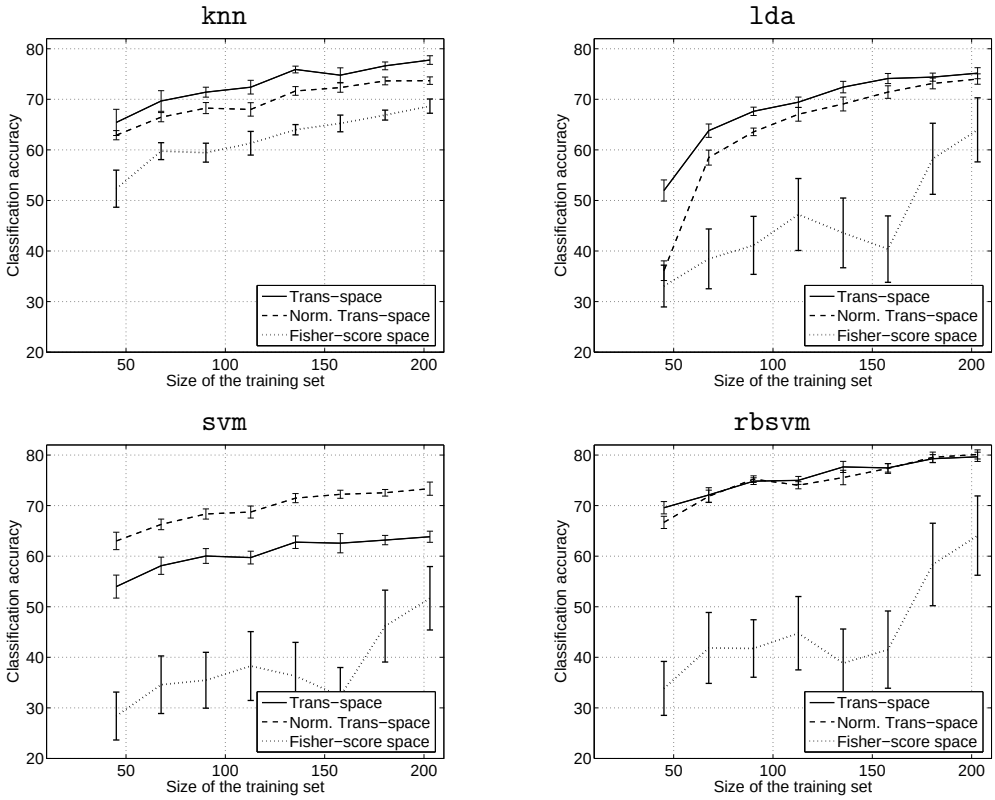

Figure 5: Scenario 1: comparison of three scaling approaches for the `trans`-space. Performance of four classifiers is evaluated as a function of the number of training examples ($T = R$), randomly selected from the design set.

where

$$\mathcal{F}_{\mathcal{CI}}^{FScore}(\mathbf{O}, \boldsymbol{\lambda}_c) = \left[\frac{\bar{\xi}_{(1,1)}(\mathbf{O}, \boldsymbol{\lambda}_c)}{a_{11}}, \ldots, \frac{\bar{\xi}_{(1,K)}(\mathbf{O}, \boldsymbol{\lambda}_c)}{a_{1K}}, \frac{\bar{\xi}_{(2,1)}(\mathbf{O}, \boldsymbol{\lambda}_c)}{a_{21}}, \ldots, \frac{\bar{\xi}_{(K,K)}(\mathbf{O}, \boldsymbol{\lambda}_c)}{a_{KK}} \right]^T . \quad (23)$$

Scaling in vector spaces is important for classifiers that depend on the distance definition, such as the k-NN rule and SVM. If the original feature space lacks a natural scaling, the scaling may be derived from the data. Fisher Scores, however, determine an automatic, natural way of scaling, as also pointed out in [29]. It is of interest to investigate whether the Fisher Scores provide a good alternative to original or normalised `trans`-space (with scaling determined from the training set). Both spaces refer to the same parameters and have equal dimensions. Such a comparison is expected to be dependent on data cardinality and should thereby be studied as a function of this size (*i.e.* by using learning curves). In particular, we compare three approaches:

1. Original `trans`-space, as defined in Section 4.1. No scaling is applied.
2. normalised `trans`-space. Features of the `trans`-space are normalised to unit variance.
3. `Fisher-Score` space: eq. (22). Features of the original `trans`-space are scaled by the corresponding a_{ij}.

Our experiments are performed on the *Chicken* data, represented by curvature sequences. Four classifiers, `lda`, `knn`, `svm` and `rbsvm`, are trained in the three corresponding HMM-induced

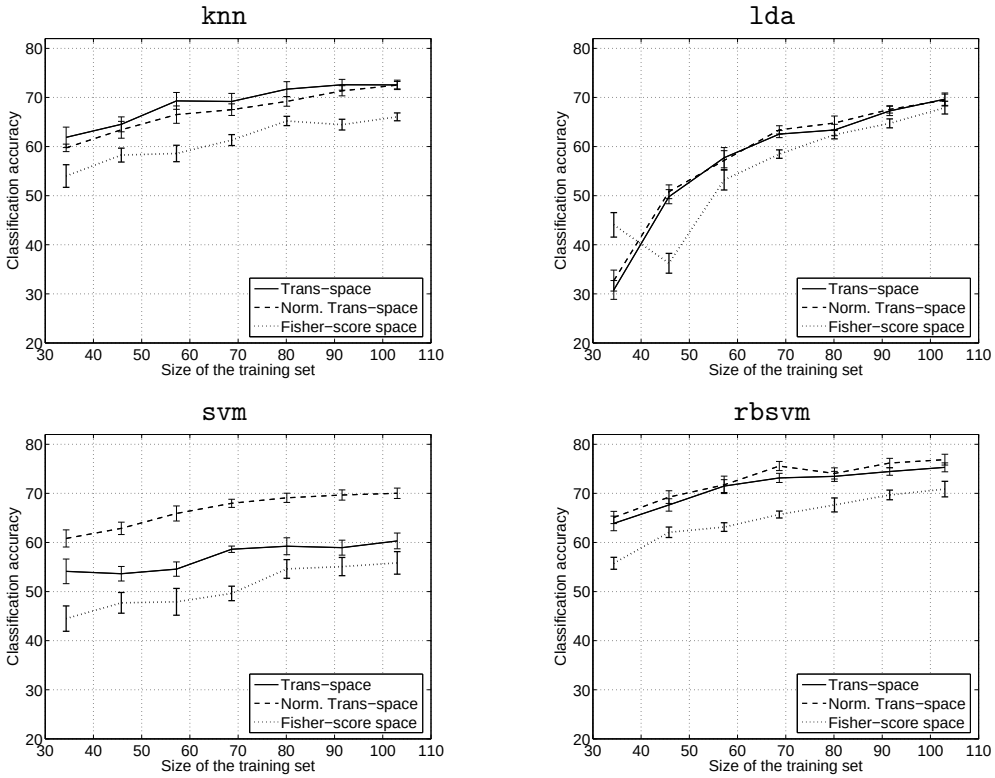

Figure 6: Scenario 2: comparison of three scaling approaches for the `trans`-space. Performance of four classifiers is evaluated as a function of the training size. The representation set R is fixed, $|R| = 0.5n_D = 111$.

vector spaces. Note that training a linear SVM in the `Fisher-Score` space is identical to the method of [29] of training SVM with the simplified Fisher kernel[10], defined as $K(\mathbf{O}_i, \mathbf{O}_j) = U_F(\mathbf{O}_i)^T U_F(\mathbf{O}_j)$.

In this study, the data set is split in the ratio 50%-50% into the *design* and test sets. The design set D consists of $n_D = 222$ examples. It serves for the construction of learning curves that show classification accuracy as a function of the number of learning sequences. Three different scenarios are considered:

- **Scenario 1:** Multiple training sets T of various sizes are randomly selected from D. Their sizes vary from $|T| = 0.1n_D$ to $|T| = 0.9n_D$. T is used in the complete learning process: to train HMMs and build the corresponding `HMMVS`, as well as to train the classifiers. The results are shown in Fig. 5.
- **Scenario 2:** A representation set R, $|R| = 0.5n_D$, is first randomly selected from D. The set R is used to train one-class HMMs. Multiple training sets T of various sizes are randomly selected from $D \backslash R$ with sizes varying from $0.1(0.5n_D)$ to $0.9(0.5n_D)$. T is used to build the HMM-induced *representation* space, `HMMVS`, and to train the classifiers there. The results are shown in Fig. 6.

[10]Fisher kernel is defined as $K(\mathbf{O}_i, \mathbf{O}_j) = U_F(\mathbf{O}_i)^T \mathcal{I}^{-1} U_F(\mathbf{O}_j)$, where $\mathcal{I}$ is the Fisher information matrix, *i.e.* the variance of the Fisher score, $\mathcal{I} = \mathrm{Var}[U_F]$. The (i,j)-element of I is $(\mathcal{I}(\boldsymbol{\alpha}))_{i,j} = \mathrm{E}\left[\frac{\partial \mathcal{L}(\boldsymbol{\alpha};\mathbf{O})}{\partial \alpha_i} \frac{\partial \mathcal{L}(\boldsymbol{\alpha};\mathbf{O})}{\partial \alpha_j}\right]$.

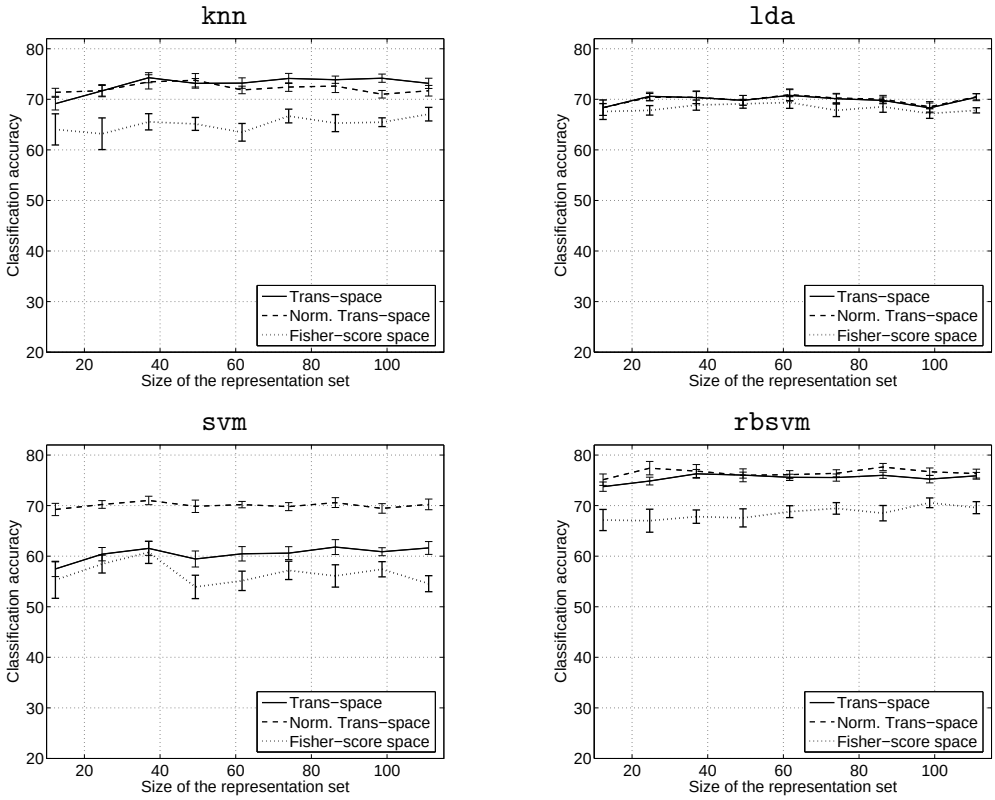

Figure 7: Scenario 3: comparison of three scaling approaches for the `trans`-space. Performance of four classifiers is evaluated as a function of the representation set size. The training set T is fixed, $|T| = 0.5n_D$.

- **Scenario 3:** A training set T consisting of $0.5n_D$ examples is first randomly selected from D. Multiple representation sets R are then randomly selected from $D \backslash T$ with sizes varying from $0.1(0.5n_D)$ to $0.9(0.5n_D)$. R is used to train one-class HMMs, while T is used to build the HMM-induced *representation* space and to train the classifiers there. The results are shown in Fig. 7.

The following observations can be made from Fig. 5 – 7. First of all, `Fisher-Score` space is inferior to `trans`-space. Normalisation of features by the corresponding transition probabilities a_{ij} is apparently bad. For small a_{ij}, the scaling of the features $\bar{\xi}_{(i,j)}(\mathbf{O}, \boldsymbol{\lambda}_c)$ by a_{ij} overemphasises the resulting values and may lead to very noisy features. The Fisher Scores improve for larger training sets thanks to more reliable estimates.

Secondly, no normalisation, *i.e.* the original `trans`-space, is usually the best choice. The natural scaling as given by (6) produces a good result for most classifiers. Only in the case of `svm`, the normalisation of features to unit variances leads to better results than in the original `trans`-space; see Fig. 5, bottom left. The results for the Fisher Scores are identical to training SVM with the Fisher kernel.

Next, the `rbsvm` tends to perform the best. This can be expected, as it uses an optimised Gaussian kernel with respect to the data. The somewhat worse performance of the two linear classifiers `lda` and `svm` indicates that the separation boundary is non-linear.

Finally, the size of the representation set R (Scenario 3), see Fig. 7, has little influence.

Apparently, ten representation sequences are sufficient to train the models; more representation sequences neither bring significantly new information, nor introduce more noise.

5.1.2 Nonlinear normalization of the score space

We have shown and confirmed the intuition made in [54] that a space normalization is often required to make the Fisher Kernel properly working. This is also true for other generative kernels. For example, the Marginalized Kernel, in its original formulation [59], needed a re-scaling of the Kernel Matrix (centering and division by the Frobenius norm) in order to perform adequately. The common characteristic of these space transformations is however the linearity of the scaling function. Nevertheless, there are situations where the linearity assumption is too restrictive, and a benefit may be obtained from a nonlinear scaling via a nonlinear mapping.

Currently, we are investigating the latter alternative, proposing a nonlinear transformation of the original vectorial space, obtained through generative embedding, into another vectorial space, namely a nonlinear mapping of space dimensions able to highlight or exploit their discriminative characteristics. In particular, we focus on a particular class of generative embeddings, namely embeddings defined on generative models with latent variables (for example, the states in a Hidden Markov Model), leading to generative kernels defined as inner product on the resulting vectorial space. A famous example of such kernels is the Marginalized Kernel [59] – even if in the original paper an explicit definition of the space and a derivation of the kernel as inner product are missing. Another example is the `state`-space presented in Sect. 4.1. The idea at the basis of generative embeddings for this class of kernels is to map the objects of the problem in a space where each dimension (or a set of dimensions) describes the contribution of one of the latent variables of the model. For example, in the `state`-space each direction represents the averaged probability of being in a particular state given the model and the observation.

In this case, we proposed to transform such spaces via a nonlinear scaling, defining new kernels. In our implementation these kernels are defined as inner products in the transformed space: the specific form of such kernels depends on the choice of the nonlinear mapping and on the latent variable model it relies upon. Diverse nonlinear mappings are indeed possible, and we investigated one possible, very simple choice of a nonlinear mapping, able to balance the contributions of each latent variable of the model, thus augmenting the entropy of the latent variables vectors. The basic tool is a powering operation, able to equilibrate the contributions of each latent variable. We apply this idea to the Marginalized Kernel and to the State-Space Kernel, giving the kernel formulation in closed form for a HMM generative model.

We are currently performing some experiments to assess the validity of the proposed approach, with really promising results. All the details will be provided in the deliverable to be issued at the end of the WP2.

References

[1] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J Mol Biol*, 215(3):403–410, October 1990.

[2] C. Bahlmann and H. Burkhardt. Measuring hmm similarity with the bayes probability of error and its application to online handwriting recognition. In *Proc. Int. Conf. Document Analysis and Recognition*, pages 406–411, 2001.

[3] Timothy L. Bailey and William Noble Grundy. Classifying proteins by family using the product of correlated p-values. In *Proceedings of the Third Annual International Conference on Computational Molecular Biology*, pages 10–14. ACM, 1999.

[4] A. Ben-Hur, D. Horn, H.T. Siegelmann, , and V. Vapnik. Support vector clustering. *Journal of Machine Learning Research*, 2:125–137, 2001.

[5] K. Bennett. Combining support vector and mathematical programming methods for induction. In B. Schölkopf, C.J.C. Burges, and A.J. Smola, editors, *Advances in Kernel Methods - SV Learning*, pages 307–326, Cambridge, MA, 1999. MIT Press.

[6] C. Bhattacharyya, L.R. Grate, A. Rizki, D. Radisky, F.J. Molina, M.I. Jordan, M.J. Bissel, and I.S. Mian. Simultaneous classification and relevant feature identification in high-dimensional spaces: Application to molecular profiling data. *Signal Processing*, 83:729–743, 2003.

[7] M. Bicego, M. Cristani, V. Murino, E. Pekalska, and R.P.W. Duin. Clustering-based construction of hidden markov models for generative kernels. In D. Cremers, Y. Boykov, A. Blake, and F. R. Schmidt, editors, *Proc. Int. Conf. on Energy Minimization Methods in Computer Vision and Pattern Recognition*, volume 5681 of *Lecture Notes in Computer Science*, pages 466–479. Springer, 2009.

[8] M. Bicego and M.A.T. Figueiredo. Soft clustering using weighted one-class support vector machines. *Pattern Recognition*, 42(1):27–32, 2009.

[9] M. Bicego and V. Murino. Investigating Hidden Markov Models' capabilities in 2D shape classification. *IEEE Trans. on Pattern Analysis and Machine Intelligence - PAMI*, 26(2):281–286, 2004.

[10] M. Bicego, V. Murino, and M.A.T. Figueiredo. Similarity-based classification of sequences using hidden markov models. *Pattern Recognition*, 37(12):2281–2291, 2004.

[11] M. Bicego, E. Pekalska, and R.P.W. Duin. Group-induced vector spaces. In M. Haindl, J. Kittler, and F. Roli, editors, *Multiple Classifier Systems*, volume LNCS 4472, pages 190–199. Springer, 2007.

[12] M. Bicego, E. Pekalska, D.M.J. Tax, and R.P.W. Duin. Component-based discriminative classification for hidden markov models. *Pattern Recognition*, in press, 2009.

[13] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[14] Matthew Brand. Structure learning in conditional probability models via an entropic prior and parameter extinction. *Neural Comput.*, 11(5):1155–1182, 1999.

[15] K. Tsuda B.Schlkopf and J. Vert. *Kernel Methods in Computational Biology.* The MIT Press, 2004.

[16] F. Camastra and A. Verri. A novel kernel method for clustering. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27:801–805, 2005.

[17] L. Chen and H. Man. Combination of fisher scores and appearance based features for face recognition. In *Proc. of ACM SIGMM Multimedia Biometrics Methods and Applications Workshop (WBMA03)*, pages 74–81, 2003.

[18] L. Chen and H. Man. Discriminant analysis of stochastic models and its application to face recognition. In *Proc. Int. Workshop on Analysis and Modeling of Faces and Gestures*, pages 5–10, 2003.

[19] L. Chen, H. man, and A.N. Nefian. Face recognition based on multi-class mapping of fisher scores. *Pattern Recognition*, 2005:799–811, 2005.

[20] M. Cuturi, K. Fukumizu, and J. Vert. Semigroup kernels on measures. *Journal of Machine Learning Research*, 6:1169–1198, 2005.

[21] I.M. de Diego, J.M. Moguerza, and A. Muñoz. Combining kernel information for support vector classification. In *Proc. Int. Conf. on Multiple Classifier Systems*, pages 102–111, 2004.

[22] R.P.W Duin, P. Juszczak, P. Paclik, E. Pekalska, D. De Ridder, and D.M.J. Tax. Prtools4, a matlab toolbox for pattern recognition, 2004. Delft University of Technology.

[23] S. Fine, J. Navratil, and R. Gopinath. A hybrid gmm/svm approach to speaker identification. In *IEEE Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 417–420, 2001.

[24] K. Fukanaga. *Introduction to Statistical pattern recognition.* Academic press, San Diego, 2nd edition, 1990.

[25] M.J.F. Gales. Discriminative models for speech recognition. In *Information Theory and Applications Workshop*, 2007.

[26] Zoubin Ghahramani. On structured variational approximations. Technical Report CRG-TR-97-1, 1997.

[27] T. Hastie, R. Tibshirani, and J.H. Friedman. *The Elements of Statistical Learning.* Springer Verlag, 2001.

[28] T. Jaakkola, M. Diekhaus, and D. Haussler. Using the fisher kernel method to detect remote protein homologies. *7th Intell. Sys. Mol. Biol.*, pages 149–158, 1999.

[29] T.S. Jaakkola and D. Haussler. Exploiting generative models in discriminative classifiers. In *Advances in Neural Information Processing Systems*, pages 487–493, 1999.

[30] A.K. Jain and R. Dubes. *Algorithms for clustering data.* Prentice Hall, 1988.

[31] T. Jebara, I. Kondor, and A. Howard. Probability product kernels. *Journal of Machine Learning Research*, 5:819–844, 2004.

[32] M.I. Jordan, Z. Ghahramani, T. Jaakkola, and L.K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37(2):183–233, 1999.

[33] H. Kappen and W. Wiegerinck. Mean field theory for graphical models, 2001.

[34] Kevin Karplus, Christian Barrett, and Richard Hughey. Hidden markov models for detecting remote protein homologies. *Bioinformatics*, 14:846–856, 1999.

[35] H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Proc. Int. Conf on Machine Learning*, 2003.

[36] C. Lai, D.M.J. Tax, R.P.W. Duin, E. Pękalska, and P. Paclík. A study on combining image representations for image classification and retrieval. *International Journal of Pattern Recognition and Artificial Intelligence*, 18(5):867–890, 2004.

[37] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:2169–2178, 2006.

[38] J. Lee and D. Lee. An improved cluster labeling method for support vector clustering. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 27(3):461–464, 2005.

[39] D. MacKay. Ensemble learning for Hidden Markov Models, 1997. Unpublished. Department of Physics, University of Cambridge.

[40] R.A. Mollineda, E. Vidal, and F. Casacuberta. Cyclic sequence alignments: Approximate versus optimal techniques. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 16(3):291–299, 2002.

[41] P.J. Moreno, P.P. Ho, and N. Vasconcelos. A kullback-leibler divergence based kernel for svm classification in multimedia applications. In *Proc. of Advances in Neural Information Processing*, volume 16, 2003.

[42] Radford M. Neal and Geoffrey E. Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. pages 355–368, 1999.

[43] M. Neuhaus and H. Bunke. Edit distance-based kernel functions for structural pattern classification. *Pattern Recognition*, 39:1852–1863, 2006.

[44] A. Ng and M. Jordan. On discriminative vs generative classifiers: A comparison of logistic regression and naive Bayes. In *Advances in Neural Information Processing Systems*, 2002.

[45] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42:145–175, 2001.

[46] A. Opelt, M. Fussenegger, A. Pinz, and P. Auer. Weak hypotheses and boosting for generic object detection and recognition. In *Proceedings of the 8th European Conference on Computer Vision*, volume 2, pages 71–84, 2004.

[47] A. Panuccio, M. Bicego, and V. Murino. A Hidden Markov model-based approach to sequential data clustering. In *Structural, Syntactic and Statistical Pattern Recognition*, volume LNCS 2396, pages 734–742. Springer, 2002.

[48] A. Perina, M. Cristani, U. Castellani, and V. Murino. A new generative feature set based on entropy distance for discriminative classification. In *Proc. Int. Conf. on Image Analysis and Processing*, pages 199–208, 2009.

[49] A. Perina, M. Cristani, U. Castellani, V. Murino, and N.Jojic. Free energy score space. In *Advances in Neural Information Processing Systems*, 2009.

[50] A. Perina, M. Cristani, U. Castellani, V. Murino, and N.Jojic. A hybrid generative/discriminative classification framework based on free-energy terms. In *Proc. Int. Conf. on Computer Vision*, 2009.

[51] L. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proc. of IEEE*, 77(2):257–286, 1989.

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

[53] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[54] N. Smith and M.J.F. Gales. Speech recognition using svms. In *Advances in Neural Information Processing Systems*, pages 1197–1204, 2002.

[55] Nathan Smith and Mark Gales. Using svms to classify variable length speech patterns. Technical Report CUED/F-INGENF/TR.412, University of Cambridge, UK, 2002.

[56] N.D. Smith. *Using Augmented Statistical Models and Score Spaces for Classification*. PhD thesis, Engineering Departement, Cambridge University, 2003.

[57] P. Smyth. Clustering sequences with hidden Markov models. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, page 648. MIT Press, 1997.

[58] Geoffrey G. Towell, Jude W. Shavlik, and Michiel O. Noordewier. Refinement of approximate domain theories by knowledge-based neural networks. In *In Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 861–866, 1990.

[59] K. Tsuda, T. Kin, and K. Asai. Marginalised kernels for biological sequences. *Bioinformatics*, 18:268–275, 2002.

[60] Koji Tsuda, Motoaki Kawanabe, Gunnar Rätsch, Sören Sonnenburg, and Klaus-Robert Müller. A new discriminative kernel from probabilistic models. *Neural Comput.*, 14(10):2397–2414, 2002.

[61] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer-Verlag, 1995.

[62] V.N. Vapnik. *Statistical Learning Theory.* Wiley, New York, 1998.

[63] Bouchard, G., & Triggs, B. (2004). The tradeoff between generative and discriminative classifiers. *IASC International Symposium on Computational Statistics* (pages 721–728). Prague.

[64] Kapadia, S. (1998). *Discriminative training of hidden markov models.* Doctoral dissertation.

[65] Mccallum, A., Pal, C., Druck, G., & Wang, X. (2006). Multi-conditional learning: Generative/discriminative training for clustering and classification. *In Proceedings of the 21st National Conference on Artificial Intelligence* (pages 433–439).