Clustering-Based Construction of Hidden Markov Models for Generative Kernels

Manuele Bicego[1,2,⋆], Marco Cristani[1,2], Vittorio Murino[1,2], Elżbieta Pękalska[3], and Robert P.W. Duin[4]

[1] Computer Science Department, University of Verona, Italy
[2] Istituto Italiano di Tecnologia (IIT), Italy
[3] School of Computer Science, University of Manchester, UK
[4] Delft University of Technology, The Netherlands

Abstract. Generative kernels represent theoretically grounded tools able to increase the capabilities of generative classification through a discriminative setting. Fisher Kernel is the first and mostly-used representative, which lies on a widely investigated mathematical background. The manufacture of a generative kernel flows down through a two-step serial pipeline. In the first, "generative" step, a generative model is trained, considering one model for class or a whole model for all the data; then, features or scores are extracted, which encode the contribution of each data point in the generative process. In the second, "discriminative" part, the scores are evaluated by a discriminative machine via a kernel, exploiting the data separability. In this paper we contribute to the first aspect, proposing a novel way to fit the class-data with the generative models, in specific, focusing on Hidden Markov Models (HMM). The idea is to perform model clustering on the unlabeled data in order to discover at best the structure of the entire sample set. Then, the label information is retrieved and generative scores are computed. Experimental, comparative test provides a preliminary idea on the goodness of the novel approach, pushing forward for further developments.

1 Introduction

Hidden Markov Models (HMMs) represent a powerful and ductile statistical learning framework. In the classical HMM-based classification a single HMM is built for each class and the Maximum A Posteriori (MAP) approach is used to classify an unlabeled sequence $\mathbf{O}$, thus following a pure generative classification scheme.

Even though the MAP rule represents the theoretically optimal decision rule (*i.e.* leading to the minimum probability of error [1]), in practice, generative classification may suffer from poor discriminative abilities. This is likely to occur in case of poorly estimated class models (e.g. due to insufficient learning examples), improper model topologies, (e.g. due to a bad model definition or conditional dependence of the states), or possible class overlap (as may occur

⋆ Corresponding author. Strada Le Grazie, 15 - 37134 Verona, Italy, Tel.: +39 045 8027072; Fax: +39 045 8027068, manuele.bicego@univr.it

D. Cremers et al. (Eds.): EMMCVPR 2009, LNCS 5681, pp. 466–479, 2009.

e.g. in medical problems where patient diagnoses vary between medical doctors). Some of these issues can be addressed by improving or/and extending classical HMMs (e.g. Hierarchical HMM [2], Factorial Hidden Markov Models [3], Coupled HMM [4] and others). Alternatively, the discriminative skills can be enhanced by training HMMs with discriminative criteria. Two popular examples are based on Maximum Mutual Information (MMI) [5] and Minimum Bayes Risk (MBR) [6], but other extensions are available. One must however remember that although discriminative criteria try to reduce the recognition error directly, they require a rather large amount of training data. Furthermore, there exist generalizations of HMMs towards probabilistic discriminative models. These are Conditional Random Fields (CRFs) [7] and Hidden CRFs (HCRFs) [8], in which conditional maximum likelihood is often used to estimate the parameters. All these discriminative techniques need complex training procedures, whereas the final classification still relies on the MAP approach.

In recent years, a new direction has aroused great interest in the Pattern Recognition community: the hybrid generative-discriminative approach. The idea is to merge the description abilities of the Hidden Markov Models (and more in general of generative models) with the discriminative skills of discriminative methods, *i.e.* methods that directly model the posterior probability and, by this, focus on the class separability. Generally speaking, there is a proven complementarity of discriminative and generative estimations: asymptotically (in the number of labeled training examples), discriminative methods lead to lower classification error than the generative ones [9], when comparing logistic regression to naive Bayes classifiers. On the other hand, generative counterparts are effective with less (and possibly unlabeled) data.

Different approaches have been proposed in this context. They may roughly be divided into two classes: generative embeddings and generative kernels. In the first case, the basic idea is to employ generative models in order to embed objects to a vectorial feature space (where any discriminative classifier can be trained – [10,11,12,13,14]). In the latter case a specific kernel is designed (which may rely either on an explicit or implicit space), further used in the Support Vector Machine scenario. Such examples can be found in [15,16,17,18]. In this paper, we will focus on the second class of approaches.

The most famous and widely investigated generative kernel, defined not only for HMM but for any generative model, is the Fisher Kernel [15], first advocated in the context of protein sequence analysis. A generative model is used to build a feature space in which a kernel is defined by suitable object comparisons. In particular, the Fisher Kernel approach measures the relation between objects by comparing them in the tangent space induced by the trained generative model. In practice, 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 [15].

In order to define a generative kernel, the first step is to employ data to build the generative model. Let us consider this problem in the Fisher Kernel case [15].

In the original scenario defined by Jaakkola and Haussler in [15], the Fisher Kernel was computed on the basis of the log-likelihood of a single generative model, representing both competing classes. Another early version, by Fine et al. [19], 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 [20] exploited such an idea in the binary classification case by proposing to employ in Fisher Kernel the derivatives of the log-ratio of the two likelihoods calculated from the two competing models. It has been shown in the paper that this scheme may enhance the discriminative power of Fisher Kernel. Other generalizations to multi-class models have been proposed, for example in [21], 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.

In this work a further contribution to the aforementioned scenario is made. The idea is to allow a generative framework a free discovery of 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), here we will explore this direction focusing on the HMM-based Fisher Kernel case, showing promising and comparative results obtained from some preliminary experiments.

The remainder of the paper is organized as follows: in Sec. 2 basic theoretical notions are provided and the notation is fixed. Sec. 3 proposes our generative kernel, whose classification comparative performances are presented in Sec. 4. Some considerations about the results are discussed in Sec.5. Finally, Sec. 6 closes the paper and opens for novel research perspectives.

2 Foundations

This section describes the basics of HMM and Fisher Kernel, mainly to fix the notation.

2.1 Hidden Markov Models

A discrete-time hidden Markov model $\boldsymbol{\lambda}$ can be viewed as a Markov model whose states are not directly observed: instead, each state is characterized by

a probability distribution function, modeling the observations corresponding to that state. More formally, an HMM is defined by the following entities [22]:

- $S = \{S_1, S_2, \cdots, S_N\}$ the finite set of possible (hidden) states;
- the transition matrix $\mathbf{A} = \{a_{ij},\ 1 \leq j \leq N\}$ representing the probability of moving from state S_i to state S_j,

$$a_{ij} = P[q_{t+1} = S_j | q_t = S_i], \quad 1 \leq i, j \leq N,$$

  with $a_{ij} \geq 0$, $\sum_{j=1}^{N} a_{ij} = 1$, and where q_t denotes the state occupied by the model at time t.
- the emission matrix $\mathbf{B} = \{b(o|S_j)\}$, indicating the probability of emission of symbol $o \in V$ when the system state is S_j; V can be a discrete alphabet or a continuous set (e.g. $V = \mathbb{R}$), in which case $b(o|S_j)$ is a probability density function.
- $\boldsymbol{\pi} = \{\pi_i\}$, the initial state probability distribution,

$$\pi_i = P[q_1 = S_i], \quad 1 \leq i \leq N$$

  with $\pi_i \geq 0$ and $\sum_{i=1}^{N} \pi_i = 1$.

For convenience, we represent an HMM by a triplet $\boldsymbol{\lambda} = (\mathbf{A}, \mathbf{B}, \boldsymbol{\pi})$.

Learning the HMM parameters, given a set of observed sequences $\{\mathbf{O}_i\}$, is usually performed using the well-known Baum-Welch algorithm [22], which is able to determine the parameters by maximizing the likelihood $P(\{\mathbf{O}_i\}|\boldsymbol{\lambda})$. One of the steps of the Baum-Welch algorithm is an evaluation step, where it is required to compute $P(\mathbf{O}|\boldsymbol{\lambda})$, given a model $\boldsymbol{\lambda}$ and a sequence $\mathbf{O}$; this can be computed using the *forward-backward procedure* [22].

2.2 Fisher Kernel

Fisher Kernel [15] was first advocated in the context of protein sequence analysis and proposed as a general way of mixing generative and discriminative models for classification. The basic idea is to employ a generative model to define feature vectors and project objects to the resulting feature space. There a meaningful similarity/distance measure is defined, leading to a kernel. In particular, the Fisher kernel approach measures the relation between the objects by comparing them in the tangent space induced by the trained generative model, which is considered as a point in the Riemannian manifold defined by a family of generative models. This space has a number of desirable characteristics, such as the possibility of measuring geodesic distances between points along the manifold (leading to the concept of natural gradients [23]). In practice, each object is represented by a feature vector, whose components are called Fisher Scores, defined by derivatives of the log-likelihood of the generative model with respect to all parameters. The dimensionality of this space equals the number of parameters. A kernel can be defined in various ways in the resulting space; the inner product was used in [15].

The general formulation is as follows: given two observations $\mathbf{O}_i$ and $\mathbf{O}_j$, and a generative model $\mathcal{P}(\mathbf{O}|\theta)$ — with θ being the vector of parameters of the generative model — Fisher Kernel is defined as:

$$FK(\mathbf{O}_i, \mathbf{O}_j) =< FS(\mathbf{O}_i, \theta), FS(\mathbf{O}_j, \theta) >$$

where $< \cdot, \cdot >$ is the inner product, and $FS(\mathbf{O}, \theta)$ is called Fisher Score and is defined as

$$FS(\mathbf{O}, \theta) = \nabla_\theta \log \mathcal{P}(\mathbf{O}|\theta)$$

In the HMM case, θ is replaced with $\boldsymbol{\lambda}$, representing the trained HMM. The vector parameter is composed by the transition probabilities, the emission probabilities (the mean and the covariance in case of Gaussian models) and the initial state probabilities. The derivation of such derivatives is not complex, and is omitted here. Interested readers are referred to [21].

3 Methodology

The construction of our HMM-based generative kernel is realized in three steps: (1) discovering the data groups, (2) building 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.

3.1 HMM-Based Clustering of Sequences

The first step is to discover natural groups in the data by performing sequence clustering. It is well known that data clustering is inherently a more difficult task than supervised classification, and this difficulty worsens if sequential data are considered: the structure of the underlying process is often difficult to infer, and typically different length sequences have to be dealt with.

The sequence clustering 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 such sense, the clustering methodology employed here is based on HMM. HMMs have not been extensively employed for clustering sequences, with only a few papers exploring this direction. More specifically, early approaches related to speech recognition were presented in [24,25,26]. A relevant contribution was made by Li and Biswas [27,28,29,30,31]). Basically, in their approach [27], the clustering problem is addressed by focusing on the model selection issue, *i.e.* the search for the HMM topology best representing data, and the clustering structure issue, *i.e.* finding the most likely number of clusters.

More advanced techniques have been proposed by Smyth [32] (see also the more general and more recent [33]), where a series of sequential steps permits to realize a block-wise HMM, modeling the whole data set. Other interesting examples can be found in [34], where HMMs are used as cluster prototypes, with the clustering obtained with the *rival penalized competitive learning* (RPCL)

algorithm, and in [35], where HMMs were employed to derive a feature space where standard vector-based clustering algorithms have been applied.

In any case, the simplest and most widely used class of approaches for HMM-based clustering is the proximity-based clustering, where the main effort of the clustering process lies in devising good similarity or distance measures between sequences. With such measures, any standard distance-based method (as agglomerative clustering) can be applied. Within this context, HMMs are employed to compute similarities between sequences, using different approaches (see for example [36,37]), and standard pairwise distance-based approaches (as agglomerative hierarchical) are then used for the final data clusters.

In our study, we chose a simple yet effective clustering method, belonging to the class of proximity-based clustering approaches. Considering a given set of N sequences $\{\mathbf{O}_1 ... \mathbf{O}_N\}$ to be clustered, the algorithm performs the following steps:

1. Train a single HMM $\boldsymbol{\lambda}_i$ for each sequence $\mathbf{O}_i$ (the details of the training are in the next section).
2. Compute the distance matrix $D = \{D(\mathbf{O}_i, \mathbf{O}_j)\}$, representing a matrix of similarities between sequences or between models. This is typically obtained either by calculating the model-likelihood probabilities, or by devising a measure of distances between models. In the past, few authors have proposed approaches to compute these distances: early approaches were based on the Euclidean distance of the discrete observation probability, while other approaches were based on entropy, or on co-emission probability of two models, or, very recently, on the Bayes probability of error (see [37] and the references therein). Here we use the following distance, employed in [32,36]. First, given the sequences $\{\mathbf{O}_j\}$ and the models $\{\boldsymbol{\lambda}_i\}$, we compute the following matrix:
$$L_{ij} = P(\mathbf{O}_j | \boldsymbol{\lambda}_i) \tag{1}$$
The similarity matrix $S(\mathbf{O}_i, \mathbf{O}_j)$ is then obtained by symmetrizing the matrix L_{ij}. Thus we define
$$S(\mathbf{O}_i, \mathbf{O}_j) = \frac{1}{2} \left[L_{ij} + L_{ji} \right]. \tag{2}$$
Clearly the choice of this distance is crucial for the effectiveness of the clustering: readers interested in this aspect may refer to [36], where different Likelihood-based distances have been considered and tested in an EEG clustering scenario.
3. Use a hierarchical agglomerative clustering method (with the Complete Link rule [38]) on $S(\mathbf{O}_i, \mathbf{O}_j)$ to perform the clustering.

Clearly the choice of the best number of clusters represents a problem, even though different indices/strategies have been proposed (see for example [38]). In our experimental evaluation we let it vary in a proper range, and report the different results.

3.2 HMM Training

Once estimated the natural groups inside the data set, a single HMM is trained for each group. HMM training is performed by using the Baum-Welch re-estimation procedure, stopping at the likelihood convergence. 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 initialization, picking the best likelihood model as the representative. In the experimental part, the best number of states has been determined with a preliminary experimental evaluation.

3.3 Kernel Computation

The goal in this phase is to compute the Fisher Kernel given a set of trained models. Here we adopt the scheme proposed in [21], and recently adopted also in [14] – where a set of different HMM-based generative embeddings have been proposed. The idea is to concatenate the scores obtained from each model. Here we adopt the same strategy, the difference is that the models are not built on the classes but on the extracted clusters.

More formally, given two sequences $\mathbf{O}_i$ and $\mathbf{O}_j$, and the set of K trained HMMs $\{\boldsymbol{\lambda}_k\}$ (with K being the number of clusters), the kernel is then determined as the inner product in the vector space being a Cartesian product of the Fisher score spaces resulting from all individual models, in the same way the Fisher Kernel is built. In other words, given a sequence $\mathbf{O}_i$, the Fisher scores $FS(\mathbf{O}_i, \boldsymbol{\lambda}_k)$, are computed using each trained model $\boldsymbol{\lambda}_k$, concatenating them in a single vector:

$$CFS(\mathbf{O}_i, \{\boldsymbol{\lambda}_k\}) = [FS(\mathbf{O}_i, \boldsymbol{\lambda}_1), FS(\mathbf{O}_i, \boldsymbol{\lambda}_2), \cdots, FS(\mathbf{O}_i, \boldsymbol{\lambda}_K)].$$

Given two concatenated vectors $CFS(\mathbf{O}_i, \{\boldsymbol{\lambda}_k\})$ and $CFS(\mathbf{O}_j, \{\boldsymbol{\lambda}_k\})$, relative to two sequences $\mathbf{O}_i$ and $\mathbf{O}_j$, the kernel is then computed as:

$$FK(\mathbf{O}_i, \mathbf{O}_j) =< CFS(\mathbf{O}_i, \{\boldsymbol{\lambda}_k\}), CFS(\mathbf{O}_j, \{\boldsymbol{\lambda}_k\}) >$$

where $< \cdot, \cdot >$ represents the inner product.

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

4 Experimental Evaluation

The proposed methodology has been tested using a 2D shape recognition problem. Recognition of 2D shapes is an unconventional application of HMMs, even though promising results have been reported [39,40,41]. The idea, in this case, is 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 [42] 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. In the first case, a standard 8-direction chain encoding procedure is applied to each image. Discrete HMMs are used to model these classes of symbol sequences. In the second case, we derive curvature sequences as in [41,43]. First, contours are extracted by using the *Canny* edge detector; the boundary is then approximated by segments of approximately fixed length. Then, at any given point x the curvature value is derived as an angle between two consecutive segments intersecting at x. The initial point is the rightmost point lying on the horizontal line passing through the object centroid, following the boundary in a counterclockwise manner. Classes of curvature sequences are finally 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 [15]. 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 [19], 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 [21], 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 [20]). 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 Tables 1 and 2, respectively.

Table 1. Continuous HMMs applied to the Chicken Database with curvature sequences: 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).

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)

Table 2. Discrete HMMs applied to the Chicken Database with chain code sequences: 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).

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)

5 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.

Moreover, the obtained results are remarkable, considering the difficulty of the data set. As a reference, we put in Table 3 some results on the same dataset: it is evident how the proposed approach performs very competitively with respect to the state of the art.

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 [20].

Table 3. Comparative Results on the Chicken dataset

Methodology	Protocol	Accuracy	Reference
1-NN + Levenshtein edit distance	Leave One Out	$\approx 67\%$	[42]
1-NN + approximated cyclic distance	Leave One Out	$\approx 78\%$	[42]
KNN + cyclic string edit distance	Train/Test/Valid	74.3%	[43]
SVM + Edit distance-based kernel	Train/Test/Validf	81.1%	[43]
1-NN + mBm-based features	Leave One Out	76.5%	[44]
1-NN + Hmm-based distance	Leave One Out	73.77%	[44]
SVM + Hmm-based entropic features	Leave One Out	81.21%	[45]

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 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 [21], 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 spaces. As 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 are independent of dimensionalities. 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. The final performance however 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 training set size.

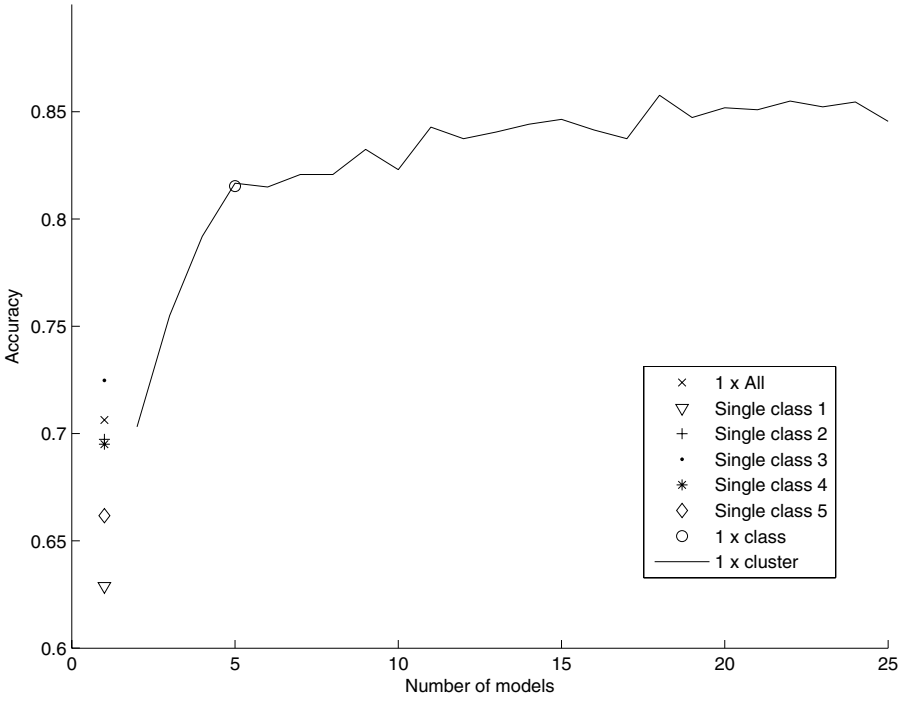

Fig. 1. Performances on chain code experiment with varying number of clusters

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 discriminated via a discriminative method.

6 Conclusions and Future Perspectives

In this paper we furnished a novel way to build a generative kernel based on Hidden Markov Models and Fisher Kernel. In the typical generative kernel building process, a generative model is fit on the data, considering class-label information: this generates two main fit directions, namely, one model for class or one model for all the data. Then, scores are extracted from the trained models which highlight the generative correspondence among points and the single model parameters. Finally, discriminative reasoning exploits data separability. Our contribution is to suggest an alternative way to build the HMM generative framework from the data. The idea is to consider all the data together, allowing possible data structural information to better emerge. This information is captured by performing model clustering, which provide few HMM models encoding all the data as surrogates of many initial simple models fit on local data points. Class label information is then recovered in the score building process, and, subsequently, by the discriminative machinery. This paper represents a preliminary work toward a novel research direction for the manufacture of generative kernels. Our work consists on several comparative tests which show what are the potentialities to take into account and the possible issues to face. The promising results allow us to further investigate this research direction.

Acknowledgements

We acknowledge financial support from the FET programme within the EU FP7, under the SIMBAD project (Contract 213250).

References

1. Duda, R., Hart, P., Stork, D.: Pattern Classification, 2nd edn. John Wiley & Sons, Chichester (2001)
2. Fine, S., Singer, Y., Tishby, N.: The hierarchical hidden markov model: Analysis and applications. Machine Learning 32, 41–62 (1998)
3. Ghahramani, Z., Jordan, M.: Factorial hidden markov models. Machine Learning 29, 245–273 (1997)
4. Brand, M., Oliver, N., Pentland, A.: Coupled hidden markov models for complex action recognition. In: Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (1997)
5. Bahl, L., Brown, P., de Souza, P., Mercer, R.: Maximum mutual information estimation of hidden Markov model parameters for speech recognition. In: IEEE International Conference on Acoustics, Speech and Signal Processing, Tokyo, Japan, vol. I, pp. 49–52 (2000)
6. Kaiser, Z., Horvat, B., Kacic, Z.: A novel loss function for the overall risk criterion based discriminative training of HMM models. In: International Conference on Spoken Language Processing, Beijing, China, vol. 2, pp. 887–890 (2000)
7. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: probabilistic models for segmenting and labelling sequence data. In: International Conference on Machine Learning, pp. 591–598 (2001)

8. Gunawardana, A., Mahajan, M., Acero, A., Platt, J.: Hidden conditional random fields for phone classication. In: Interspeech, Lisbon, Portugal, pp. 1117–1120 (2005)
9. Ng, A., Jordan, M.: On discriminative vs generative classifiers: A comparison of logistic regression and naive Bayes. In: Advances in Neural Information Processing Systems (2002)
10. Bicego, M., Murino, V., Figueiredo, M.: Similarity-based classification of sequences using hidden markov models. Pattern Recognition 37(12), 2281–2291 (2004)
11. Bicego, M., Pękalska, E., Duin, R.P.W.: Group-induced vector spaces. In: Haindl, M., Kittler, J., Roli, F. (eds.) MCS 2007. LNCS, vol. 4472, pp. 190–199. Springer, Heidelberg (2007)
12. Layton, M., Gales, M.: Augmented statistical models: Exploiting generative models in discriminative classifiers. In: Advances in Neural Information Processing Systems (2005)
13. Smith, N.: Using Augmented Statistical Models and Score Spaces for Classification. PhD thesis, Engineering Departement, Cambridge University (2003)
14. Bicego, M., Pekalska, E., Tax, D., Duin, R.: Component-based discriminative classification for hidden markov models. Pattern Recognition (in press, 2009)
15. Jaakkola, T., Haussler, D.: Exploiting generative models in discriminative classifiers. In: Advances in Neural Information Processing Systems, pp. 487–493 (1999)
16. Tsuda, K., Kin, T., Asai, K.: Marginalised kernels for biological sequences. Bioinformatics 18, 268–275 (2002)
17. Jebara, T., Kondor, I., Howard, A.: Probability product kernels. Journal of Machine Learning Research 5, 819–844 (2004)
18. Moreno, P., Ho, P., Vasconcelos, N.: A kullback-leibler divergence based kernel for svm classification in multimedia applications. In: Proc. of Advances in Neural Information Processing., vol. 16 (2003)
19. Fine, S., Navratil, J., Gopinath, R.: A hybrid gmm/svm approach to speaker identification. In: IEEE Int. Conf. on Acoustics, Speech, and Signal Processing, pp. 417–420 (2001)
20. Smith, N., Gales, M.: Speech recognition using svms. In: Advances in Neural Information Processing Systems, pp. 1197–1204 (2002)
21. Chen, L., Man, H., Nefian, A.: Face recognition based on multi-class mapping of fisher scores. Pattern Recognition, 799–811 (2005)
22. Rabiner, L.: A tutorial on Hidden Markov Models and selected applications in speech recognition. Proc. of IEEE 77(2), 257–286 (1989)
23. Amari, S.: Natural gradient works efficiently in learning. Neural Computation 10, 251–276 (1998)
24. Rabiner, L., Lee, C., Juang, B., Wilpon, J.: HMM clustering for connected word recognition. In: Proc. Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP), pp. 405–408 (1989)
25. Lee, K.: Context-dependent phonetic hidden Markov models for speaker-independent continuous speech recognition. IEEE Transactions on Acoustics, Speech and Signal Processing 38(4), 599–609 (1990)
26. Kosaka, T., Matsunaga, S., Kuraoka, M.: Speaker-independent phone modeling based on speaker-dependent hmm's composition and clustering. In: Int. Proc. on Acoustics, Speech, and Signal Processing, vol. 1, pp. 441–444 (1995)
27. Li, C.: A Bayesian Approach to Temporal Data Clustering using Hidden Markov Model Methodology. PhD thesis, Vanderbilt University (2000)

28. Li, C., Biswas, G.: Clustering sequence data using hidden Markov model representation. In: Proc. of SPIE 1999 Conf. on Data Mining and Knowledge Discovery: Theory, Tools, and Technology, pp. 14–21 (1999)
29. Li, C., Biswas, G.: A bayesian approach to temporal data clustering using hidden Markov models. In: Proc. Int. Conf. on Machine Learning, pp. 543–550 (2000)
30. Li, C., Biswas, G.: Applying the Hidden Markov Model methodology for unsupervised learning of temporal data. Int. Journal of Knowledge-based Intelligent Engineering Systems 6(3), 152–160 (2002)
31. Li, C., Biswas, G., Dale, M., Dale, P.: Matryoshka: A HMM based temporal data clustering methodology for modeling system dynamics. Intelligent Data Analysis Journal (2002)
32. Smyth, P.: Clustering sequences with hidden Markov models. In: Mozer, M., Jordan, M., Petsche, T. (eds.) Advances in Neural Information Processing Systems, vol. 9, p. 648. MIT Press, Cambridge (1997)
33. Cadez, I., Gaffney, S., Smyth, P.: A general probabilistic framework for clustering individuals. In: Proc. of ACM SIGKDD 2000 (2000)
34. Law, M., Kwok, J.: Rival penalized competitive learning for model-based sequence. In: Proc. Int. Conf. Pattern Recognition, vol. 2, pp. 195–198 (2000)
35. Bicego, M., Murino, V., Figueiredo, M.: Similarity-based clustering of sequences using hidden Markov models. In: Perner, P., Rosenfeld, A. (eds.) MLDM 2003. LNCS (LNAI), vol. 2734, pp. 86–95. Springer, Heidelberg (2003)
36. Panuccio, A., Bicego, M., Murino, V.: A hidden markov model-based approach to sequential data clustering. In: Caelli, T.M., Amin, A., Duin, R.P.W., Kamel, M.S., de Ridder, D. (eds.) SPR 2002 and SSPR 2002. LNCS, vol. 2396, pp. 734–742. Springer, Heidelberg (2002)
37. Bahlmann, C., Burkhardt, H.: Measuring hmm similarity with the bayes probability of error and its application to online handwriting recognition. In: Proc. Int. Conf. Document Analysis and Recognition, pp. 406–411 (2001)
38. Jain, A., Dubes, R.: Algorithms for clustering data. Prentice-Hall, Englewood Cliffs (1988)
39. He, Y., Kundu, A.: 2-D shape classification using Hidden Markov Model. IEEE Trans. Pattern Analysis Machine Intelligence 13(11), 1172–1184 (1991)
40. Arica, N., Yarman-Vural, F.: A shape descriptor based on circular Hidden Markov Model. In: IEEE Proc. Int Conf. Pattern Recognition, vol. 1, pp. 924–927 (2000)
41. Bicego, M., Murino, V.: Investigating Hidden Markov Models' capabilities in 2D shape classification. IEEE Trans. on Pattern Analysis and Machine Intelligence - PAMI 26(2), 281–286 (2004)
42. Mollineda, R., Vidal, E., Casacuberta, F.: Cyclic sequence alignments: Approximate versus optimal techniques. Int. Journal of Pattern Recognition and Artificial Intelligence 16(3), 291–299 (2002)
43. Neuhaus, M., Bunke, H.: Edit distance-based kernel functions for structural pattern classification. Pattern Recognition 39, 1852–1863 (2006)
44. Bicego, M., Trudda, A.: 2D shape classification using multifractional brownian motion. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) S+SSPR 2008. LNCS, vol. 5342, pp. 906–916. Springer, Heidelberg (2008)
45. Perina, A., Cristani, M., Castellani, U., Murino, V.: A new generative feature set based on entropy distance for discriminative classification. In: Proc. of Int. Conf. on Image Analysis and Processing, ICIAP 2009 (2009)