A new generative feature set based on entropy distance for discriminative classification

A. Perina[1], M. Cristani[1,2], U.Castellani[1], and V.Murino[1,2]

[1] Dipartimento di Informatica,
Università degli Studi di Verona,
Strada le Grazie 15, 37134 Verona, Italia.
[2] Istituto Italiano di Tecnologia,
Via Morego 30, 16163 Genova, Italia.
[alessandro.perina,marco.cristani,umberto.castellani,vittorio.murino]@univr.it

Abstract. Score functions induced by generative models extract fixed-dimensions feature vectors from different-length data observations by subsuming the process of data generation, projecting them in highly informative spaces called score spaces. In this way, standard discriminative classifiers such as support vector machines, or logistic regressors are proved to achieve higher performances than a solely generative or discriminative approach. In this paper, we present a novel score space that capture the generative process encoding it in an entropic feature vector. In this way, both uncertainty in the generative model learning step and "local" compliance of data observations with respect to the generative process can be represented. The proposed score space is presented for hidden Markov models and mixture of gaussian and is experimentally validated on standard benchmark datasets; moreover it can be applied to any generative model. Results show how it achieves compelling classification accuracies.

1 Introduction

Pursuing principled hybrid architectures of discriminative and generative classifiers is currently one of the most interesting, useful, and difficult challenges for Machine Learning [1–4]. The underlying motivation is the proved complementarity of discriminative and generative estimations: asymptotically[1] classification error of discriminative methods is lower than for generative ones [5]. On the other side, generative counterparts are effective with less, possibly unlabeled, data; further, they provide intuitive mappings among structure of the model and data features.

Among these methods, "generative score space" approaches grow in the recent years their importance in the literature. Firstly one usually has to learn an estimate of the parameters θ of the generative model; we refer with $\hat{\theta}$ to this estimate. Secondly, a score function $\phi(\mathbf{x}, \hat{\theta})$ is defined in order to map samples

[1] In the number of labeled training examples

$\mathbf{x}$ in a fixed-size dimensional space built using the estimate of the parameters [1, 2]. After the mapping, a space metric must be defined in order to employ discriminative approaches.
Most of the literature on this topic exploits the hidden Markov models as generative frameworks, but any other generative methods can be employed [4].
In the literature, the most known score space is the Fisher space [2], in which a single model for all the classes or a set of per-class models are fit on the data. Then the tangent vector of the marginal log likelihood $\nabla_\theta \log p(x|\hat{\theta})$ is used as feature vector; the Fisher Kernel refers simply to the inner product in this space.
In general, the main intuition of generative score space approaches is to distill the contribution of each parameter θ_i in the generation of a particular sample.
Similarly to [2], other score spaces have been proposed, like the Top Kernel [1] derived from $\nabla_\theta(\log p(y = +1|x, \hat{\theta}) - \log p(y = -1|x, \hat{\theta}))^2$, the tangent vector of posterior log-odds, or the Fisher space variants presented in [3, 4].
Following this trend, in this paper we propose a novel feature extractor, introduced here for hidden Markov models, that extracts features from a learned generative model, moving the generative description of the data into a set of features that can be used in a discriminative framework. The idea is to capture the difference in the generative process between the samples, calculating a distance between the statistics collected over all the samples, encoded in the parameter estimate $\hat{\theta}$, and the individual j-th sample statistics $p(\mathbf{y}|x^{(j)})$, where $\mathbf{y}$ represent the set of hidden variables, possibly containing the class variable. The intuition is that the samples of a particular class have to differ in the same way (i.e., have to present the same generative behavior) from the parameters estimate $\hat{\theta}$.
The rest of the paper is organized as follows. In Sec. 2 technical preliminaries are reported. In Sec. 3, the proposed framework is introduced and several generalizations are discussed in Sec. 4. An exhaustive experimental section is presented in Sec. 5, and, conclusions are drawn in Sec. 6.

2 Hidden Markov models

Hidden Markov models (HMMs) are generative models aimed at modeling sequential data. As visible in Figure 1, they are composed by a sequence of hidden state variables $\mathbf{S} = \{S_k\}_{k=1}^K$ and by a sequence of visible variables $\mathbf{O} = \{O_k\}_{k=1}^K$. In the case of a first-order hidden Markov model, each state variable S_k depends on the previous state variable S_{k-1} via a conditional dependency, and influences the visible observation variable O_k. Each value of k identifies a *slice* of the model. More formally, a HMM $\boldsymbol{\lambda}$ is defined by the number hidden states Q, and by the following parameters [6]:

1. A transition matrix $\mathbf{A} = \{a_{mn}\}$, $1 \leq m, n \leq Q$ representing the probability of moving from state m to state n,

$$a_{mn} = P(S_{k+1} = n|S_k = m), \quad 1 \leq n, m \leq Q, \quad \forall k = 1, \ldots, K$$

[2] y is the label variable

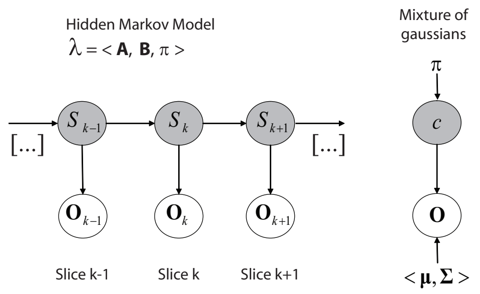

Fig. 1. On the left an hidden Markov model $\boldsymbol{\lambda} =< \mathbf{A}, \mathbf{B}, \pi >$. On the right the graphical model that represent the mixture of Gaussians which can be viewed as a single slice of an hidden Markov model with gaussian emission.

2. An emission matrix $\mathbf{B} = \{b_m(v)\}$, indicating the probability of emission of symbol $v \in V$ when the system state is m; V can be a discrete alphabet or a continuous set (e.g. $V = I\!R$) and, in this case, $b_m(v)$ is a probability density function.
3. $\boldsymbol{\pi} = \{\pi_m\}$, the initial state probability distribution, $\pi_m = P(S_1 = m)$

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

The learning problem Given a model $\boldsymbol{\lambda}$ and a set of training observation sequences $\{\mathbf{O}^{(j)}\}_{j=1}^{J}$, the goal of a learning algorithm is to choose the values of the parameters $\boldsymbol{\lambda}$ that maximize the data likelihood $P(\{\mathbf{O}^{(j)}\}_{j=1}^{J}|\boldsymbol{\lambda})$. The procedure that exploits the model learning is a generalization of the well known Expectation-Maximization algorithm, known as Baum-Welch procedure [6].
The E-step consists in first calculating the standard forward and backward variables α and β using the *forwards-backwards procedure* [6]. From these variables key quantities can be obtained, such as the conditional probability of two consecutive hidden states in an observation sequence at site k, *i.e.*, $P(S_k = m, S_{k+1} = n|\mathbf{O}^{(j)}) = \xi_k^j(m,n)$ and the conditional $P(S_k = m|\mathbf{O}^{(j)})= \sum_{n=1}^{L} \xi_k^j(m,n) = \gamma_k^j(m)$. In the M-step the prior distribution π and the transition $\mathbf{A}$ and the emission $\mathbf{B}$ matrices and are updated using these quantities.

3 Entropy features for hidden Markov models

Given C hidden Markov models $\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_C$, to perform generative classification of a sequence $\mathbf{O}^{(j)}$, one has to solve the evaluation problem for each class $c = 1, \ldots, C$ obtaining $P(\mathbf{O}^{(j)}|\boldsymbol{\lambda}_c)$, and then assign the most likely class label $\hat{c}$, after computing the posterior distribution $P(c|\mathbf{O}^{(j)})$ via Bayes rule (*Maximum-a-posteriori* (MAP) classification). Nevertheless, as stated in [7], generative classification often yields to poor accuracies if compared with discriminative methods. To get the best from both frameworks, an alternative hybrid framework has been

proposed: the idea is to exploit the generative process in a discriminative framework, trying to extract features from a generative model previously learned.
Here, we focus on hidden Markov models, and unlike generative classification, we want to move the focus from likelihoods, to the different behaviors of the samples under the generative model. To do this we learn a single hidden Markov model pooling all the samples of all C classes together, trying to capture the different generative process for each pair of examples $\mathbf{O}^{(j)}$ and $\mathbf{O}^{(h)}$ looking at their posterior distributions. The intuition is that the samples of the different classes have to "use" different paths over the hidden Markov models states S_k in order to fit the model parameters θ.
To catch this difference we use, as feature for a discriminative classifier, the distance between the posterior distribution of each sample $P(\mathbf{S}|\mathbf{O}^{(j)}, \boldsymbol{\lambda})$ and the joint distribution $P(\mathbf{S}, \mathbf{O}|\boldsymbol{\lambda})$ which is calculated collecting statistics over all the samples. In this way we can see *where* the samples of a class differentiate from the others.
Since we are calculating a distance between two distributions, we employ the entropy distance, defined as $\mathcal{H}(p, q) = -p \log q$; therefore the final feature vector becomes

$$\begin{aligned}
\mathcal{H} &= -P(\mathbf{S}|\mathbf{O}^{(j)}, \boldsymbol{\lambda}) \log P(\mathbf{S}, \mathbf{O}|\boldsymbol{\lambda}) \\
&= -\sum_{[S_k]} P(S_k|\mathbf{O}^{(j)}, \boldsymbol{\lambda}) \log \left[\pi_1 \cdot \left(\prod_{k=2}^{K} a_{s_k, s_{k-1}} \right) \cdot \left(\prod_{k=1}^{K} b_{s_k}(v) \right) \right] \\
&= -\sum_{m=1}^{Q} P(S_1 = m|\mathbf{O}^{(j)}, \boldsymbol{\lambda}) \log \pi_m - \sum_{k=1}^{K} \sum_{m=1}^{Q} \sum_{v} P(S_k = m|\mathbf{O}^{(j)}, \boldsymbol{\lambda}) \log b_m(v) \\
&\quad - \sum_{k=2}^{K} \sum_{m,n=1}^{Q} P(S_k = m, S_{k+1} = n|\mathbf{O}^{(j)}, \boldsymbol{\lambda}) \log a_{m,n} \qquad (1)
\end{aligned}$$

Instead of calculating the sums present in equation 1, we keep all the addends separated in order to have many descriptive features. In this way, for each j-th sample, we can extract a feature vector using the following three step procedure.

Transition probabilities The statistics collected over all the sequences are encoded in the transition matrix $P(S_{k+1} = n|S_k = m) = a_{mn}$, which represent the probability of transition between states n and m. The same quantity, referred to a single sample j, is the probability that it moves from the state m to state n, i.e. $P(S_k = m, S_{k+1} = n|\mathbf{O}^{(j)}, \boldsymbol{\lambda}) = \xi_k(m, n)$. In this way we can calculate a entropic distance term for each slice k, for each couple of states m, n as

$$\mathcal{H}_t(m, n, k) = -\xi_k(m, n) \cdot \log a_{m,n} \qquad (2)$$

Emission probabilities The probability that the hidden Markov model emits a symbol v being in the state m is $b_m(v)$, the statistics collected over a single sample is the probability that the sequence is in the state m if the current symbol is v, otherwise is 0. Therefore the distance between the generative process of a sample referring to the emission process and the emission matrix, can be calculated for each state m, for each symbol v and for each slice k as

$$\mathcal{H}_e(m, v, k) = -\gamma_k(m) \cdot \delta_{O_k^{(j)},v} \cdot \log b_m(v) \tag{3}$$

where $\delta_{i,j}$ is the Kronecker delta.

Prior Probability Similarly we can calculate a distance vector for each state value m between the sample posterior and the prior state distribution as

$$\mathcal{H}_p(m) = -\gamma_1(m) \cdot \log \pi_m \tag{4}$$

These distances can be concatenated and used as a feature vector for discriminative classification, since they encode the generative process that created the examples. Finally, we define the feature extractor ϕ that maps an observation $\mathbf{O}^{(j)}$ into a vector of entropy distances as:

$$\phi(\mathbf{O}^{(j)}, \mathbf{HMM}) : \mathbf{O}^{(j)} \rightarrow [\overbrace{\ldots, \mathcal{H}_e(m, v, k), \ldots}^{Q \cdot V \cdot K \;\; long}, \overbrace{\ldots, \mathcal{H}_t(m, n, k), \ldots}^{Q \cdot Q \cdot K \;\; long}, \overbrace{\ldots, \mathcal{H}_p(m), \ldots}^{Q \;\; long}] \tag{5}$$

The classification protocol is simple, an HMM $\boldsymbol{\lambda}$ is learned using the training data, then the extractor ϕ is applied to both training and testing data obtaining the data for the discriminative classifier.

4 Generalizations: variable length sequences and other models

The major problem that (space) projection methods solve effectively is the classification of variable length sequences, such as audio recordings, DNA strings or shapes. Using the proposed feature extractor ϕ, each sequence is mapped in a space whose dimension depends on the length of the sample (see equations 2-3). In fact, if the j-th sample has length $K(j)$, the feature vector would be $Q \cdot K(j) \cdot (Q + V) + Q$ long, making it unusable for discriminative methods.
To solve this problem, we can simply sum over the slices $k = 1 \ldots K(j)$. Note how this operation is eligible and natural, since each piece of equations 2-4 is an entropy and so it is their sum. Therefore we define a second set of features as:

$$\overline{\mathcal{H}}_e(m, v) = -\sum_k \mathcal{H}_e(m, v, k), \quad \overline{\mathcal{H}}_t(m, n) = -\sum_k \mathcal{H}_t(m, n, k) \tag{6}$$

consequently, the mapping operator ϕ would result in

$$\phi(\mathbf{O}^{(j)}, \mathbf{HMM}) : \mathbf{O}^{(j)} \rightarrow [\overbrace{\ldots, \overline{\mathcal{H}}_e(m, v), \ldots}^{Q \cdot V \;\; long}, \overbrace{\ldots, \overline{\mathcal{H}}_t(m, n, t), \ldots}^{Q \cdot Q \;\; long}, \overbrace{\ldots, \mathcal{H}_p(m), \ldots}^{Q \;\; long}] \tag{7}$$

This framework can be further generalized to any generative model that contains a mixture variable that can separate the behaviors of the samples. Probably the most famous and simple generative model with such characteristics is the mixture of gaussians (MOG), which can be thought as a single slice of an hidden Markov model (see Figure 1).
Given a set of samples $\mathbf{X} = \left\{\mathbf{X}^{(j)}\right\}_{j=1}^{J}$, the joint distribution of a mixture of gaussians is

$$P(C,\mathbf{X}) = \sum_c P(C=c)\cdot P(\mathbf{X}|C=c) = \prod_{j=1}^{J}\left(\sum_c \pi_c \cdot \mathcal{N}(\mathbf{X}^{(j)};\mu_c,\Sigma_c)\right) \quad (8)$$

where C is the number of the mixture components. For each sample, in the E-step, the responsibilities $P(C=c|\mathbf{X}^{(j)})$ are calculated, then these statistics are collected to calculate the means μ_c and the covariance matrices Σ_c (M-step).
At this point, the derivation of the entropy features can be easily written as:

$$\mathcal{H}_\mu(c) = -P(C=c|\mathbf{X}^{(j)})\cdot\log\mathcal{N}(\mathbf{X}^{(j)};\mu_c,\Sigma_c) \qquad \mathcal{H}_p(c) = -P(C=c|\mathbf{X}^{(j)})\cdot\log\pi_c \quad (9)$$

leading to the following feature extractor operator:

$$\phi(\mathbf{X}^{(j)},\mathbf{MOG}) : x^t \rightarrow [\overbrace{\ldots,\mathcal{H}_\mu(c),\ldots,}^{C\ \ long}\overbrace{\ldots,\mathcal{H}_p(c),\ldots}^{C\ \ long}] \quad (10)$$

5 Experiments

Having the feature set $\phi(\mathbf{X},\cdot)$ been extracted, as discriminative classifier we employ support vector machines with gaussian RBF kernel [8] even though any other discriminative classifier could be employed. The choice of the particular kernel used, is guided by the fact that the feature vectors contain entropies, whose distance is usually calculated using the $\mathbb{L}_2$ norm [9].

5.1 Mixture of gaussians

These experiments aim to show how the proposed features bring generative information to the discriminative classifiers boosting the classification accuracies. We compared the results with the MAP estimate of the generative model used to extract the features (**MOG**), with support vector machines (**SVM**) with RBF kernel, using directly the data as feature, and with an SVM on the proposed features ($\phi(\mathbf{X},\mathbf{MOG})$ + **SVM**). For each experiment we learned a mixture of gaussians from the training data with C components[3], where C is the number of the classes of the particular dataset.
The first dataset is the Fisher's *Iris* dataset [10], perhaps the best known dataset

[3] The number of classes in a classification task is a-priori known, so we do not have to investigate on C.

in the pattern recognition. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant.
The second dataset is the *Wine* dataset [10]. Data points are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivations. The analysis determined the quantities of 13 constituents found in each of the 3 types of wines.
The *Blood* dataset [10] is composed by 748 samples described by 5 features. The associated task is to predict if samples had their blood drawn on a particular date (2 classes).
To perform generative classification, we learned a separate mixture of gaussian ($C = 2$) for each class, and we classify testing data using the Bayes rule. Results

Mixture of Gaussians

Dataset	**Wine**	**Iris**	**Blood**
Classes	3	3	2
MOG (MAP)	73,0%	87,1%	69,9%
SVM	83,1%	97,1%	76,3%
$\phi(\mathbf{X}, \mathbf{MOG})$ + **SVM**	**86,6%**	**98,9%**	**78,3%**

Table 1. Mixture of gaussians numerical results. All the improvements are statistically significant however the main benefit of the method is that differently from **SVM**, it can deal with missing data or multiple length sequences (see Sec. 5.2).

for all datasets are shown in Table 1 and confirm that discriminative methods outperform generative classification, but the same discriminative method (SVM) obtains better performances when used with the proposed features since the mapping $\phi(\mathbf{X}, \mathbf{MOG})$ additionally encodes the generative process that created the data.

5.2 Hidden Markov models

To evaluate our approach using HMMs, we focused on some standard datasets considering as comparative results, the classification accuracies reported by the dataset's authors, the generative classification based on likelihoods (**HMM**) and the Fisher and TOP Kernels [1, 2] in their original definition (indicated respectively with **FK** and **TK**).
The number of states of the hidden Markov model Q has been chosen using a validation set extracted from the training set and the same HMM used for generative classification is used for the extraction of the Fisher, Top and Entropy features.
In the first test we generated 800 fixed-length sequences, sampling 8 hidden Markov models (100 sequences per class). We classified the sequences using the 50% of the data for training and the rest for testing (50-50), repeating the process 10 times. The mean results are reported in table 2, where one can note that the improvement with respect to **HMM**, **FK** and **TK** is evident and statistically significant.

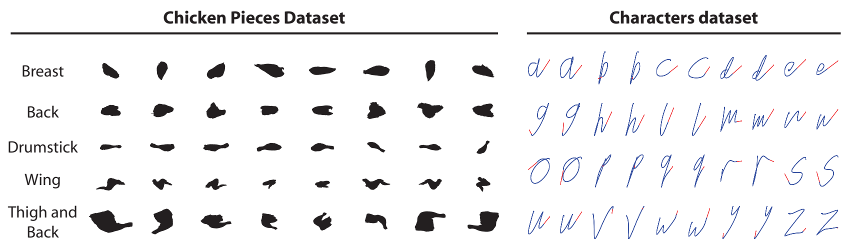

Fig. 2. On the left, examples from each of the 5 classes of the Chicken Pieces database. On the right examples from the Characters database.

The first dataset is the Chicken Pieces Database[4] [11], which consists of 446 binary images of chicken pieces, belonging to one of the five classes that represent specific parts of a chicken: "wing", "back", "drumstick", "thigh and back" and "breast". Despite the limited number of classes, this is a challenging dataset where the best result do not go over 81% of accuracy, at best of our knowledge. The shapes are usually first described by contours, which are further encoded by suitable sequences; this makes the classification task even more difficult. In order to compute curvature sequences, the contours are first extracted by using the Canny edge detector, and the boundary is then approximated by segments of approximately fixed length. Finally, the curvature value at point x is computed as the angle between the two consecutive segments intersecting at x, resulting in continuous valued sequences of different length. Some images from this dataset are depicted in Figure 2.
Results published in [11] report a baseline leave-one-out accuracy of 67% by using the 1-NN on the Levenshtein (non-cyclic) edit distance computed on the contour chain code. In [12] the authors characterize the contour of each object using the multifractional Brownian motion (mBm), using Horst coefficients to derive a fixed length vector, which characterizes each shape. After that a 1-NN classifiers (with the Euclidean and the Minkowsky distance) is used. In [13] a kernel based method, based on a dissimilarity representation, was proposed. Note that each of these three methods employs a different technique to deal with variable length sequences. Moreover, we took as comparison the score space methods presented in [1, 2][5].
We compared our method with [11, 12] using leave-one-out (LOO) and with [1, 2, 13] using 50% of the data for training and the rest for testing (50-50), repeating the process 10 times. Results are reported in Table 2, confusion matrices are depicted in Figure 3; the proposed method strongly outperforms all the comparisons used.

The second dataset is the characters dataset used for a Ph.D. study on primitive extraction using HMM-based models [10, 14]. The data consists of 2858

[4] `http://algoval.essex.ac.uk:8080/data/sequence/chicken`
[5] A brief description of the two methods can be found in Section 1

character samples, divided in 20 classes (see Figure 2 for some example). Data was captured using a WACOM tablet. Three dimensions were kept: x, y, and pen tip force. Such data, captured at 200Hz, has been numerically differentiated and smoothed using a Gaussian with a sigma value equal to 2. This process results in a set of 3-dimensional continuous observations.
Classification has never been performed on this data, except some preliminary result over a restricted subset (p vs q).
Results are summarized in Table 2; also in this case the proposed approach outperforms the Fisher kernel [2], and generative classification with statistical significance. Top Kernel obtains slightly superior accuracy than the proposed method, but we cannot claim statistical difference between the two results. In Figure 3 we reported the confusion matrices.

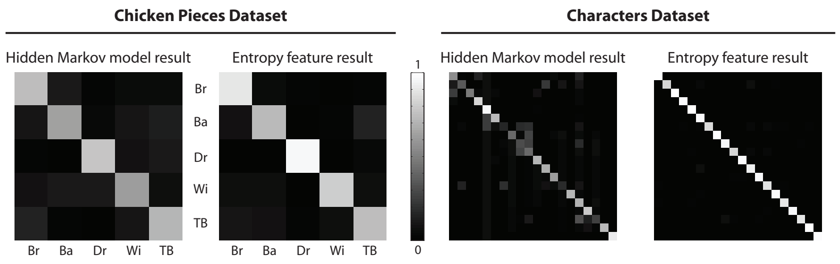

Fig. 3. Confusion matrices for generative hidden Markov classification and for the proposed approach for the chicken and the character datasets.

Hidden Markov models

Dataset	**Synthetic**	**Chicken pieces**		**Characters**
Feature - Classes	Continuous - 8	Curvature - 5		WACOM tablet - 20
Validation	50-50	50-50	LOO	50-50
$\phi(\mathbf{O}, \mathbf{HMM})$ + **SVM**	**88,43%**	**80,80%**	**81,21%**	**92,91%**
HMM (MAP)	71,31%	68,31%	-	57,30%
FK [2]	81,19%	79,12%	-	89,26%
TK [1]	82,95%	78,11%	-	**93,67%**
[11]		n.a.	$\approx$ 67%	
[12]		n.a.	76,5%	
[13]		74,3%	n.a.	

Table 2. Hidden Markov Models numerical results for synthetic, chicken and character datasets. Differently from **MOG** experiments, discriminative methods cannot be applied directly here since the input sequences have different lengths. Best results are highlighted using bold numbers if they are statistically significant.

6 Conclusions

In this paper, we presented a novel generative feature set based on the differences in generative process. The features obtained by means of the proposed mapping

ϕ, encode ambiguity within the generative model, resulting from sub-optimal learning, but, at the same time, they mirror discrepancies of the test samples with respect to the generative data process. An large experimental section shows the goodness of the approach and exhibits more than promising performances. In almost all the tests performed our method achieves the best score, outperforming with a large margin, the generative model upon which the proposed feature are extracted. Moreover the tests show how the proposed method can successfully deal with either fixed or variable length data of different natures.

Acknowledgments

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

Bibliography

[1] Tsuda, K., Kawanabe, M., Rätsch, G., Sonnenburg, S., Müller, K.R.: A new discriminative kernel from probabilistic models. Neural Comput. **14** (2002) 2397–2414

[2] Jaakkola, T., Haussler, D.: Exploiting generative models in discriminative classifiers. NIPS (1998)

[3] Smith, N., Gales, M.: Using svms to classify variable length speech patterns. Technical Report CUED/F-INGENF/TR.412, University of Cambridge, UK (2002)

[4] Holub, A.D., Welling, M., Perona, P.: Combining generative models and fisher kernels for object recognition. ICCV **1** (2005) 136–143

[5] Ng, A.Y., Jordan, M.I.: On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In Dietterich, T.G., Becker, S., Ghahramani, Z., eds.: NIPS, Cambridge, MA, MIT Press (2002)

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

[7] Liang, P., Jordan, M.I.: An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. In: ICML '08: Proceedings of the 25th international conference on Machine learning, New York, NY, USA, ACM (2008) 584–591

[8] Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. The MIT Press (2001)

[9] Akhloufi, M., Larbi, W.B., Maldague, X.: Framework for color-texture classification in machine vision inspection of industrial products. In: ISIC. (2007) 1067–1071

[10] Asuncion, A., Newman, D.J.: UCI machine learning repository (2007)

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

[12] Bicego, M., Trudda, A.: 2d shape classification using multifractional brownian motion. In: SSPR/SPR. (2008) 906–916

[13] Neuhaus, M., Bunke, H.: Edit distance-based kernel functions for structural pattern classification. Pattern Recogn. **39** (2006) 1852–1863

[14] Williams, B.H., Toussaint, M., Storkey, A.J.: Extracting motion primitives from natural handwriting data. In: ICANN (2). (2006) 634–643