2011 10th International Conference on Machine Learning and Applications

Metric Learning for Music Symbol Recognition

Ana Rebelo, *Student Member, IEEE*, Jakub Tkaczuk, Ricardo Sousa, *Student Member, IEEE*, Jaime S. Cardoso, *Member, IEEE*
INESC Porto, Faculdade Engenharia, Universidade Porto
Campus da FEUP, Rua Dr. Roberto Frias, n. 378
4200-465 Porto, Portugal
{arebelo,jakub.tkaczuk,rsousa,jaime.cardoso}@inescporto.pt

***Abstract*—Although Optical Music Recognition (OMR) has been the focus of much research for decades, the processing of handwritten musical scores is not yet satisfactory. The efforts made to find robust symbol representations and learning methodologies have not found a similar quality in the learning of the dissimilarity concept. Simple Euclidean distances are often used to measure dissimilarity between different examples. However, such distances do not necessarily yield the best performance.**

In this paper, we propose to learn the best distance for the k-nearest neighbor (k-NN) classifier. The distance concept will be tuned both for the application domain and the adopted representation for the music symbols. The performance of the method is compared with the support vector machine (SVM) classifier using both real and synthetic music scores. The synthetic database includes four types of deformations inducing variability in the printed musical symbols which exist in handwritten music sheets. The work presented here can open new research paths towards a novel automatic musical symbols recognition module for handwritten scores.

I. Introduction

Music has been shared and remembered by aural transmission, common to folk and popular musical genres, and in the form of written documents normally called musical scores. Prior to music typographical systems, all music was copied manually including large scores and each and every part for the players and singers. Publishers have typeset a large body of historical musical scores, principally from what is known as classical music. However, there remains a substantial and important corpus of works that exist as original handwritten manuscripts (or facsimiles of these manuscripts such as photocopies). The problem is not restricted to historical documents: many contemporary compositions also exist only in handwritten or facsimile format. These important cultural artifacts are in danger of being lost through the normal damages caused by time. Some form of typesetting, or, ideally, a computer system capable of automatically decode the symbolic images and create new scores is required to preserve the music (rather than the documents themselves). Programs for optical music recognition (OMR) have been under thorough expansion for many years; however, the results still need improvement.

An OMR system has three main objectives: the recognition, the representation and the storage of musical scores in a machine-readable format. Hence, an OMR program should be able to detect the musical content and to interpret each musical symbol of a musical work. The output format containing all the musical information should be easily understandable by a computer.

A typical OMR framework for the automatic recognition of a set of music sheets has three principal modules after the image preprocessing (see Fig. 1): (1) recognition of musical symbols; (2) reconstruction of the musical information in order to build a logical description of musical notation; and (3) construction of a musical notation model to be represented as a symbolic description of the musical sheet. The first module is commonly subdivided into three stages: staff lines detection and removal to obtain an image containing only the musical symbols; symbols primitives segmentation and recognition. In the Musical Notation Reconstruction module the symbols' primitives are merged to form musical symbols. Graphical and syntactic rules can be used to introduce context information to validate and solve problems that can occur on the previous module of music symbol recognition. Detected symbols are interpreted and assigned a musical meaning. In the third module of final representation construction, a format of musical description is created with the information previously produced.

The focus of this paper is in the symbol classification step on the music symbol recognition stage. More specifically, we will learn a Mahalanobis distance for k-nearest neighbor (k-NN) classification applied to music symbol recognition.

An overview of the existent works in the classification area for music symbols is addressed in Section II followed by a brief background knowledge overview in Section III. The metric learning process is described in Section IV. In Section V we present the metric learning in the OMR context using k-NN and in Section VI we extend it to Support Vector Machines (SVMs). In Section VII, the dataset adopted and the experimental results obtained in this comparative study are presented. Finally, conclusions are drawn and future work is outlined in Section VIII.

II. Related Works

The recognition of musical primitives is often preceded with the detection and elimination of staff lines and with the detection and extraction of the music symbols.

The operation of symbol classification is in many works linked with the process of segmenting the objects from the music score [2], [3]. For the pattern recognition phase, many problems result from the difficulty in obtaining individual

978-0-7695-4607-0/11 $26.00 © 2011 IEEE
DOI 10.1109/ICMLA.2011.94

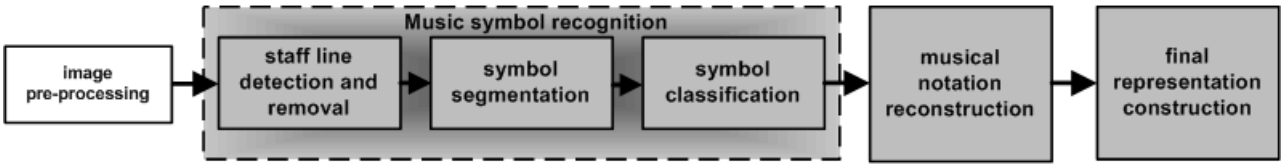

Fig. 1. Typical architecture of an OMR processing system. From [1, Fig.1].

meaningful objects. This is typically due to the printing and digitalization, as well as the paper degradation over time. In addition, distortions caused by staff lines, broken and overlapping symbols, differences in sizes and shapes or zones of high density of symbols, contribute to the complexity of the operation.

One of the techniques used to isolate the musical symbols is related to the the bounding box size, the number and organization of their constituent sections [4]. Other authors [5], [6] have chosen to apply projections to detect symbols' primitives. The recognition is done using features extracted from the projection profiles. In [5], the k-nearest neighbor rule is used in the classification phase, while neural networks is the classifier selected in [6]. The extraction of symbol features, such as width, height, area, number of holes and low-order central moments were proposed by [7]. Taubman et al. [8] preferred to extract standard moments, centralized moments, normalized moments and Hu moments. Both systems classify the music primitives using the k-nearest neighbor method.

Rossant [3] proposed a fuzzy model supported on a robust symbol detection and template matching for the extraction and recognition symbols. The method also incorporates graphical and syntactic rules in an effort to improve robustness. A structural method based on the construction of graphs for each symbol is adopted by [9]. Coüasnon [10] suggested a recognition process entirely controlled by grammar, which formalizes the musical knowledge. In [11] the segmentation and classification were performed simultaneously using Hidden Markov Models (HMMs).

Decision trees and clustering methods were used in [12] to recognize music notation. For the experimental comparison, the symbols were distorted by noise, printing defects, different fonts, skew and curvature of scanning. In [1] SVMs, neural networks (NNs), k-NN and HMMs were applied for music symbols classification. Elastic deformation technique [13] was applied to the original music symbols in order to augment the training dataset.

Other methods use classifiers with reject option [14]. The technique integrates in the classification model a confidence measure in order to reject uncertain patterns namely broken and touching symbols. The advantage of this approach is mainly the opportunity to label critical items for manual revision, instead of trying to automatically classify every item.

For the best of our knowledge there are no conducted experiments using metric learning in music symbols classification. We believe that taking advantage of distance learning can significantly improve the classification accuracy. The learnt distance metric will be directly connected with the application domain and the adopted symbol representation. Armed with this metric, the classifier has the potential to achieve a better recognition task.

III. Background Knowledge

Classifiers are built by taking a set of labeled examples that are used to construct a rule that will assign a label to any new example. In other words, if we consider a general situation, we will have a training data set $\{\mathbf{x}_i, y_i\}$, where $\mathbf{x}_i$ is a feature vector for an object i, and y_i is the label associated with the object class. The relative costs of mislabeling each $\mathbf{x}_i$ are known and must be used to generate a decision criterion that can take any new observation vector $\mathbf{x}_j$ and assign it a class label $\hat{y}_j$.

A. Support Vector Machines

One of the most widely adopted techniques by the pattern recognition community is the Support Vector Machines (SVM). This procedure has as its main idea the margin maximization having an hyperplane as the decision surface [15]. More formally, given the training set $\{\mathbf{x}_i, y_i\}_{i=1}^N$ with input data $\mathbf{x}_i \in \mathbf{R}^p$ and corresponding binary class labels $y_i \in \{-1, 1\}$, the maximum-margin hyperplane is defined by $g(\mathbf{x}) = \mathbf{w}^t\varphi(\mathbf{x}) + b$ where $\varphi(\mathbf{x})$ denotes a fixed-feature space transformation and b a bias parameter; $\mathbf{x}$ is assigned to class 1 if $g(\mathbf{x}) > 0$ or to -1 if $g(\mathbf{x}) < 0$. The maximization of the margin is equivalent to solving

$$\begin{aligned} &\min_{w,b,C,\xi_i} \quad \frac{1}{2}\mathbf{w}^t\mathbf{w} + C\sum_{i=1}^{N}\xi_i \\ &s.t \begin{cases} y_i[\mathbf{w}^t\varphi(\mathbf{x}) + b] \geq 1 - \xi_i, & i = 1, \ldots, N \\ \xi_i \geq 0 \end{cases} \end{aligned} \tag{1}$$

where parameter $C > 0$ controls the trade-off between the classification errors and the margin. The slack variables ξ_i, $i = 1, \ldots, N$ are introduced to penalyze incorrectly classified data points. The dual formulation in Equation (1) leads to a dependence on the data only through inner-products $\phi(\mathbf{x}_i)^t\phi(\mathbf{x}_j)$. Mercer's theorem allows us to express those inner products as a continuous, symmetric, positive semi-definite kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$ defined in the input space. In this work, a radial-basis function kernel was used, given by $k(\mathbf{x}, \mathbf{x}_i) = \exp(-\gamma\|\mathbf{x} - \mathbf{x}_i\|^2), \gamma \geq 0$. The binary SVM classifier can be extended to multiclass scenarios. Of the multiple extensions available in the literature [16], we used the one against one methodology.

B. K-Nearest Neighbor

The k-nearest neighbor algorithm is amongst the simplest of all machine learning algorithms [17], [18]. This method belongs to a set of techniques called Instance-based Learning. It requires no training effort and critically depends on the quality of the distances measures among each instances. It uses the heuristic that sample points near an unclassified point should indicate the class of that point. In this manner, a k-nearest neighbor classifier finds the k data points from the training set closest to the point being considered, and classifies the object with the most frequent class amongst its k-nearest neighbors.

IV. Metric Learning

Recently, the distance metric problem has received much attention in the machine learning community [19]–[21]. The performance of all machine learning algorithms depend critically on the metric that is used over input space. Some learning algorithms, such as K-means and k-nearest neighbors, require a metric that will reflect important relationships between each classes in data and will allow to discriminate instances belonging to one class from others. Depending on the availability of training examples, distance metric learning algorithms can be divided into two main categories: supervised distance metric learning and unsupervised distance metric learning. Supervised distance metric learning can be further divided into global distance metric learning and local distance metric learning. In global case constrains are applied to all pairwise of examples, while in local approach only local pairwise are taken into consideration.

A. Metric Learning for k-NN

As already mentioned previously, in the k-nearest neighbor algorithm the decision about classifying new query is determined by the labels of the k training examples with shortest distance. Conventionally those distances are defined by the Euclidean distance between examples. Decision rule classifies unlabeled inputs by the majority label of their k-nearest neighbor in the training set.

Our approach is based on work conducted by [21]. The authors proposed a distance metric learning algorithm for Large Margin Nearest Neighbor classification (LMNN). The main idea behind is to learn a Mahalanobis distance function by minimizing an objective function that is set up with local and global constraints. This optimization results in bringing k-nearest neighbors from the same class closer (i.e. shrinks the distances between nearby examples from the same class) and to separate examples from other classes by a large margin (expands the distances between examples from different classes). We introduce the idea of LMNN as follows:

Let the $\{\mathbf{x}_i, y_i\}_{i=1}^{N}$ denotes a training set of N labeled examples with inputs $\mathbf{x}_i \in \mathbf{R}^p$ and discrete class labels y_i. The Authors also introduced a binary matrix $\tau_{ij} \in \{0, 1\}$, which indicates wherever or not the labels y_i and y_j match. The main goal is to learn a linear transformation L, which will optimize k-NN classification: $L : \mathbf{R}^d \to \mathbf{R}^d$. Transformation L is used to calculate squared distances as:

$$D(\mathbf{x}_i, \mathbf{x}_j) = \|L(\mathbf{x}_i - \mathbf{x}_j)\|^2 \tag{2}$$

For each input $\mathbf{x}_i$ authors introduced k other inputs, called *target neighbors*, that share the same label y_i. These target neighbors after transformation will have minimal distance to $\mathbf{x}_i$. In the situation where any prior knowledge is not available, target inputs can be identified as k-nearest neighbor, determined by the well known Euclidean distance. Authors also introduced $\eta_{ij} \in \{0, 1\}$ in order to indicate whether $\mathbf{x}_j$ is a target neighbor of $\mathbf{x}_i$. Both matrices τ_{ij} and η_{ij} are fixed during learning stage. Formally, the cost function is defined as:

$$\epsilon(L) = \sum_{ij} \eta_{ij} \|L(\mathbf{x}_i - \mathbf{x}_j)\|^2 + c \sum_{ijl} \eta_{ij}(1 - \tau_{il}) \left[1 + \|L(\mathbf{x}_i - \mathbf{x}_j)\|^2 - \|L(\mathbf{x}_i - \mathbf{x}_l)\|^2\right]_+ \tag{3}$$

where $[z]_+ = \max(z, 0)$ denotes the standard hinge loss and $c > 0$ is a positive constant. There are two competing terms in this equation. The first one penalizes large distances between each input and its target neighbors. The second term penalizes small distances between each input and all other inputs that do not share the same label. For each input $\mathbf{x}_i$, the hinge loss in incurred by differently labeled inputs by one absolute unit of distance. The learning idea behind this approach is presented on Fig. 2.

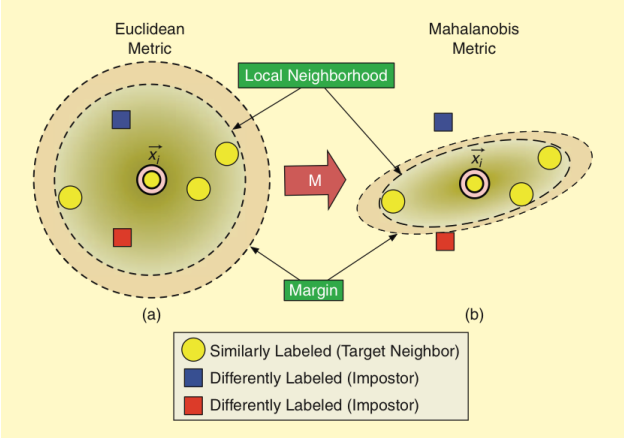

Fig. 2. Schematic illustration idea behind LMNN. Left side shows traditional approach, under Euclidean distance, where $\mathbf{x}_i$ has three target neighbors. Right image shows the new discriminator after the Mahalanobis distance being learnt. From [22, Fig.1].

The Equation (3) can be reformulated as an instance of semidefinite programming (SDP). As SDPs are convex (linear costs and constraints are replaced by convex costs and constraints) the global minimum can be efficiently computed [23]. In order to obtain SDP Equation (2) needs to be rewritten as:

$$D(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i - \mathbf{x}_j)^T M (\mathbf{x}_i - \mathbf{x}_j) \tag{4}$$

where the matrix $M = L^T L$ parametrizes the Mahalanobis distance induced by the linear transform L. In order to *mimick*

the hinge loss, slack variables ξ_{ij} were introduced for all pairs of differently labeled inputs. Finally, the resulting SDP is given by:

$$\min \sum_{ij} \eta_{ij}(\mathbf{x}_i - \mathbf{x}_j)^T M(\mathbf{x}_i - \mathbf{x}_j) + c \sum_{ij} \eta ij(1 - \tau_{il})\xi_{ijl}$$
$$s.t \begin{cases} z \geq 1 - \xi_{ijl} \\ \xi_{ijl} \geq 0 \\ M \succeq 0 \end{cases} \tag{5}$$

where $z = (\mathbf{x}_i - \mathbf{x}_l)^T M(\mathbf{x}_i - \mathbf{x}_l) - (\mathbf{x}_i - \mathbf{x}_j)^T M(\mathbf{x}_i - \mathbf{x}_j)$.

V. Metric Learning for KNN in OMR

Most authors use margin-based multi-class classification method, for instance SVMs, as a benchmark for classifying music scores. This is a common approach and can be identified in the state of the art. However, it is possible to find strong similarities between LMNN and SVMs [21]. The competing terms in Equation (3) are analogues to these present in cost function at SVMs. One term penalizes the norm of the *parameter* vector (linear transformation in distance metric, or (in SVM) the weight vector of the maximum margin hyperplane). The second terms are responsible for hinge loss over the examples that violate the condition of unit margin (the goal of margin maximization and a convex objective function are based on hinge loss). Moreover, LMNN has no explicit dependence on the number of classes, while in SVMs (multi-class classification) the training time scales at least linearly in the number of classes. For these reasons we decided to perform comparative study between LMNN and SVMs.

For this comparative study, each image of a symbol was initially resized to 20×20 pixels and then 7 music symbol features were extracted for each image. The extracted seven features were based on the Gamera project[1] and were the following:

1) the percentage of black pixels in the 20×20 pixels window of the image;
2) the orientation of the symbol;
3) the number of vertical holes;
4) the number of horizontal holes;
5) the compactness (the ratio between volume and connected components area);
6) the number of end points in the object skeleton;
7) the number of intersections in the object skeleton.

A Blurred Shape Model (BSM) descriptor [24] was also used and added to the previous features. The aim was to increase the final performance of the classifier by including characteristics that distinguish similar objects. A BSM descriptor encodes the probability of pixel densities of image regions and hence symbols are described by a probability density function. Through the high gradient magnitude of the pixels the shape of the symbol can be codified in terms of a set of key points. Then a grid comprised by a set of spatial regions is defined by the BSM descriptor. In the end, the spatial relations among key points from neighbor regions are established and features are computed. The output descriptor is a vector histogram where each position represents a distribution of probabilities of the symbol structure considering spatial distortions encompassing four possible sizes: 8×8, 16×16, 32×32 and 64×64 [24]. We opted for a feature vector histogram of the size 16×16, not only due to the computational effort, but also because a higher definition grid would not provide a richer information (contours, structure, etc) due to the already working image size (20×20).

[1] http://gamera.informatik.hsnr.de

VI. Metric Learning for SVM

An intuitive extension of the metric learning on k-NN described in this manuscript is its application to SVMs. The kernel trick, i.e., the mapping of patterns of the input feature space to a higher dimensional space, can be considered as a (dis)similarity measure. Thus, kernels can be seen as a way of a general metric learning approach [25], [26].

In this work we apply the concept introduced in [22] in order to assess the benefit of LMNN to SVMs on OMR. Our approach consisted on the strategy one-against-one where a matrix L is learnt for each discriminant. We used a kernel derived from the RBF presented in Equation (2) which resulted in $D(\mathbf{x}, \mathbf{x}_i) = \exp(-\gamma \|L(\mathbf{x} - \mathbf{x}_i)\|^2), \gamma \geq 0$, dRBF.

VII. Experimental Testing

A data set of both real handwritten scores and synthetic scores was adopted to perform the comparative study between LMNN and SVMs. The real scores consist on a set of 65 handwritten scores from 6 different composers. Images were previously binarized with the Otsu threshold algorithm. In the synthetic data set, a number of distortions were applied. This set consists on the fraction of the dataset, available from [27], written on the standard notation. The deformations applied to these printed scores were curvature, rotation, Kanungo and white speckles – see [27] for more details. In total, 380 distorted images were generated from 19 original scores.

The relevant classes for handwritten/printed music symbols used in the training phase of the classification models are presented in Table I. The symbols are grouped according to their shape. The rests symbols were divided into two groups – RestI and RestII. In total the classifiers were evaluated on a database containing 7128 examples divided into 20 classes.

For evaluation of the pattern recognition processes, the available dataset was randomly split into three sub-sets: training, validation and test sets, with 25%, 25% and 50% of the data, respectively. This division was repeated 20 times in order to obtain more stable results for accuracy by averaging and also to assess the variability of this measure. No special constraint was imposed on the distribution of the categories of symbols over the training, validation and test sets; we only guaranteed that at least one example of each category was present in the training set. The best parametrization of each model was found using the training and validation sets being the expected error estimated on the test set by a 4-cross validation scheme. In this manner, for SVM classifier C and γ values were obtained based on a grid search. In LMNN

Music Symbols										
	Accent	 BassClef	Beam	 Flat	 Natural	Note	 NoteFlag	NoteOpen	RestI	RestII
	Sharp	 TimeN	TrebleClef	TimeL	AltoClef	Relation	Breve	Semibreve	Dots	Barlines

TABLE I
FULL SET OF HANDWRITTEN AND PRINTED MUSIC SYMBOLS CONSIDERED.

algorithm the same grid search was also conducted in order to obtain the optimal value of the nearest similar labeled vectors. A confidence interval as the one presented in [1] was estimated for the mean of the error of the model on the test set.

In this work the different classifiers were tested using different sets of features extracted: 7, 16, and 23 features. Tables II and III present the results obtained applying LMNN and SVMs classifiers in the OMR database. The accuracy rates were compared using Euclidean and Mahalanobis distances. According with our expectations using information about the metric improved the results of prediction error. The first assessment is that LMNN achieved the best results where the highest gain was obtained using 23 features with LMNN against k-NN. Moreover, within SVM methodology, an overall improvement on using the metric learning on SVM can be stated with exception for the set using the 23 features. Even though existing a continuous improvement by consistently adding new features when using dRBF, one can assess that the overall performance is higher when using 23 features with the standard RBF. By not performing a cross validation on k in dRBF (due to time constrains) could be the reason behind this behavior. Furthermore, since we first optimize a distance during the metric learning phase which will be used to construct the kernel SVM, resulting thus in a similarity kernel, this transformation could also produce performance losses.

VIII. CONCLUSION

In this article a distance metric learning was successfully applied in the k-NN classifier to recognize music symbols. The results achieved in our experiments showed an improvement in comparison with k-NN with the simply Euclidean distance. We have also applied this method to derive a RBF SVM kernel (dRBF) which provided significant improvements. Recent works have focused in the application of metric learning in more advanced classifiers than k-NN, which is the classical and the simplest method for pattern recognition. Nguyen and Guo [26] proposed a metric learning support vector machine (MLSVM) method, where the problem of metric learning is formulated as a quadratic SDP problem for local neighbors constraints. Notwithstanding, k-NN is still the most basic application for metric learning, because we can easily demonstrate that with a proper distance metric, the process can improve the accuracy. In the future, we are planning to adapt other metric learning algorithms[2], for instance Probabilistic Global Distance Metric Learning (PGDM) or Active Distance Metric Learning, to other classification methods.

[2]http://www.cs.cmu.edu/~liuy/distlearn.htm

ACKNOWLEDGMENT

This work was partially supported by Fundação para a Ciência e a Tecnologia (FCT) - Portugal through project SFRH/BD/60359/2009. The third author would also like to thank Fundação para a Ciência Tecnologia (FCT) - Portugal for the Financial support through project PTDC/EIA/64914/2006.

REFERENCES

[1] A. Rebelo, G. Capela, and J. S. Cardoso, "Optical recognition of music symbols: A comparative study," *International Journal on Document Analysis and Recognition*, vol. 13, pp. 19–31, 2010.

[2] J. S. Cardoso, A. Capela, A. Rebelo, C. Guedes, and J. P. da Costa, "Staff detection with stable paths," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 6, pp. 1134–1139, 2009.

[3] F. Rossant and I. Bloch, "Robust and adaptive omr system including fuzzy modeling, fusion of musical rules, and possible error detection," *EURASIP Journal on Advances in Signal Processing*, vol. 2007, no. 1, pp. 160–160, 2007.

[4] D. Blostein and H. S. Baird, "A critical survey of music image analysis," in *Structured Document Image Analysis*, Baird, Bunke, and Yamamoto, Eds. Heidelberg: Springer-Verlag, 1992, pp. 405–434.

[5] I. Fujinaga, "Staff detection and removal," in *Visual Perception of Music Notation: On-Line and Off-Line Recognition*, S. George, Ed. Idea Group Inc., 2004, pp. 1–39.

[6] P. Bellini, I. Bruno, and P. Nesi, "Optical music sheet segmentation," *Proceedings of the 1st International Conference on Web Delivering of Music*, pp. 183–190, Nov. 2001.

[7] G. S. Choudhury, M. Droetboom, T. DiLauro, I. Fujinaga, and B. Harrington, "Optical music recognition system within a large-scale digitization project," in *International Society for Music Information Retrieval (ISMIR 2000)*, 2000.

[8] G. Taubman, A. Odest, and C. Jenkins, "Musichand: A handwritten music recognition system," Tech. Rep., 2005.

[9] R. Randriamahefa, J. Cocquerez, C. Fluhr, F. Pepin, and S. Philipp, "Printed music recognition," *Proceedings of the Second International Conference on Document Analysis and Recognition*, pp. 898–901, Oct 1993.

[10] B. Coüasnon, "Segmentation et reconnaissance de documents guidées par la connaissance a priori: application aux partitions musicales," Ph.D. dissertation, Universit de Rennes, 1996.

[11] L. Pugin, "Optical music recognition of early typographic prints using Hidden Markov Models," in *International Society for Music Information Retrieval (ISMIR)*, 2006, pp. 53–56.

[12] W. Homenda and M. Luckner, "Automatic knowledge acquisition: Recognizing music notation with methods of centroids and classifications trees," in *IJCNN*, 2006, pp. 3382–3388.

[13] A. K. Jain and D. Zongker, "Representation and recognition of handwritten digits using deformable templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 12, pp. 1386–1391, 1997.

[14] C. Dalitz, "Reject options and confidence measures for knn classifiers," in *Schriftenreihe des Fachbereichs Elektrotechnik und Informatik Hochschule Niederrhein*, C. Dalitz, Ed. Shaker Verlag, 2009, vol. 8, pp. 16–38.

[15] V. N. Vapnik, *Statistical Learning Theory*. Wiley-Interscience, September 1998.

[16] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines," *Neural Networks, IEEE Transactions on*, vol. 13, no. 2, pp. 415–425, Mar 2002.

	k-NN			LMNN		
	7	16	23	7	16	23
Accent	63%	73%	71%	66%	73%	77%
AltoClef	80%	95%	86%	81%	94%	88%
Barlines	77%	50%	84%	78%	50%	88%
BassClef	70%	89%	75%	72%	88%	85%
Beams	78%	75%	85%	78%	73%	87%
Breve	100%	100%	100%	100%	100%	100%
Dots	94%	94%	94%	94%	94%	96%
Flat	77%	92%	83%	79%	92%	87%
Naturals	82%	92%	86%	83%	93%	91%
Notes	66%	85%	73%	68%	83%	78%
NotesFlags	46%	83%	49%	76%	80%	59%
NoteOpen	71%	92%	76%	75%	92%	81%
Relation	66%	74%	71%	70%	74%	77%
RestsI	67%	84%	73%	71%	86%	78%
RestsII	70%	73%	75%	71%	72%	80%
Semibreve	71%	71%	73%	74%	71%	77%
Sharps	84%	86%	88%	85%	86%	93%
TimeL	57%	81%	69%	60%	81%	79%
TimeN	49%	71%	56%	52%	71%	65%
TrebleClef	81%	97%	82%	83%	96%	87%
99% CI for the Expected performance in percentage: average (standard deviation)	[72 (0.2); 72 (0.8)]	[80 (0.5); 82 (1.9)]	[77 (0.0); 78 (0.0)]	[73 (0.3); 74 (1.0)]	[80 (0.6); 82 (2.2)]	[82 (0.0); 83 (0.0)]

TABLE II
ACCURACY OBTAINED WITH DIFFERENT INPUT DATA: VECTOR OF 7 FEATURES, VECTOR OF 16 FEATURES AND VECTOR OF 23 FEATURES.

	SVM (RBF)			SVM (dRBF,$k = 1$)		
	7	16	23	7	16	23
Accent	51%	54%	79%	67%	63%	72%
AltoClef	79%	77%	92%	84%	89%	90%
Barlines	79%	70%	90%	79%	81%	87%
BassClef	69%	78%	82%	74%	86%	80%
Beams	82%	73%	89%	81%	74%	87%
Breve	100%	00%	100%	100%	100%	100%
Dots	95%	94%	98%	95%	93%	97%
Flat	60%	90%	90%	75%	90%	86%
Naturals	76%	90%	93%	81%	88%	86%
Notes	63%	67%	81%	67%	77%	76%
NotesFlags	40%	52%	63%	46%	74%	60%
NotesOpen	59%	57%	80%	67%	85%	75%
Relation	67%	50%	78%	65%	71%	70%
RestsI	52%	73%	74%	65%	82%	70%
RestsII	76%	65%	88%	71%	68%	78%
Semibreve	65%	0%	73%	69%	65%	79%
Sharps	87%	68%	93%	82%	81%	88%
TimeL	57%	67%	84%	58%	67%	70%
TimeN	63%	64%	75%	54%	70%	69%
TrebleClef	87%	87%	91%	82%	87%	87%
99% CI for the Expected performance in percentage: average (standard deviation)	[70 (0.2); 71 (0.7)]	[68 (0.3); 69 (1.0)]	[85 (0.2); 85 (0.7)]	[72 (0.4); 73 (1.6)]	[78 (0.3); 79 (1.2)]	[79 (0.4); 81 (1.3)]

TABLE III
ACCURACY OBTAINED WITH DIFFERENT INPUT DATA: VECTOR OF 7 FEATURES, VECTOR OF 16 FEATURES AND VECTOR OF 23 FEATURES.

[17] T. Cover and P. Hart, "Nearest neighbor pattern classification," *Information Theory, IEEE Transactions on*, vol. 13, no. 1, pp. 21 – 27, jan 1967.

[18] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition)*. Wiley-Interscience, November 2000.

[19] Z. Zhang, J. T. Kwok, and D.-Y. Yeung, "Parametric distance metric learning with label information," in *Proceedings of the 18th international joint conference on Artificial intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003, pp. 1450–1452.

[20] L. Yang and R. Jin, "Distance metric learning: A comprehensive survey," Department of Computer Science and Engineering, Michigan State University, Tech. Rep., 2006.

[21] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, pp. 207–244, June 2009.

[22] K. Weinberger, F. Sha, and L. Saul, "Convex optimizations for distance metric learning and pattern classification [applications corner]," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 146–158, may 2010.

[23] L. Vandenberghe and S. Boyd, "Semidefinite programming," *SIAM Review*, vol. 38, pp. 49–95, 1996.

[24] S. Escalera, A. Fornés, O. Pujol, P. Radeva, G. Sánchez, and J. Lladós, "Blurred shape model for binary and grey-level symbol recognition," *Pattern Recognition Letters*, vol. 30, pp. 1424–1433, November 2009.

[25] B. Schlkopf, "The kernel trick for distances," in *Advances in neural information processing systems 13*, 1993, pp. 5–3.

[26] N. Nguyen and Y. Guo, "Metric learning: A support vector approach," in *Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II*, ser. ECML PKDD '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 125–136.

[27] C. Dalitz, M. Droettboom, B. Czerwinski, and I. Fujigana, "A comparative study of staff removal algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 753–766, 2008.