Pattern Recognition 47 (2014) 1031–1038

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Separability versus prototypicality in handwritten word-image retrieval

Jean-Paul van Oosten*, Lambert Schomaker

Department of Artificial Intelligence, University of Groningen, PO Box 407, 9700 AK, Groningen, The Netherlands

ARTICLE INFO

Available online 20 September 2013

Keywords:
Image retrieval
Handwriting recognition
Nearest centroid
Support-vector machines
Separability
Prototypicality
Historical manuscripts
Big data
Continuous machine learning

ABSTRACT

Hit lists are at the core of retrieval systems. The top ranks are important, especially if user feedback is used to train the system. Analysis of hit lists revealed counter-intuitive instances in the top ranks for good classifiers. In this study, we propose that two functions need to be optimised: (a) in order to reduce a massive set of instances to a likely subset among ten thousand or more classes, separability is required. However, the results need to be intuitive after ranking, reflecting (b) the prototypicality of instances. By optimising these requirements sequentially, the number of distracting images is strongly reduced, followed by nearest-centroid based instance ranking that retains an intuitive (low-edit distance) ranking. We show that in handwritten word-image retrieval, precision improvements of up to 35 percentage points can be achieved, yielding up to 100% top hit precision and 99% top-7 precision in data sets with 84 000 instances, while maintaining high recall performances. The method is conveniently implemented in a massive scale, continuously trainable retrieval engine, Monk.

1. Introduction

In handwriting recognition, classification is often performed using statistical methods [1,2]. The class indexed *i* with the highest posterior probability given the sample to be classified is chosen as the result of the classifier

$$\text{Hypothesis}_X = \arg\max_i P(C_i|X) \quad \text{where } i \in \{1, N_{\text{classes}}\} \tag{1}$$

However, when the goal is word search, rather than automatic text transcription, the user is more interested in retrieval of word instances. Instead of a single classification, the result is a sorted hit list *H*. Each instance indexed *j* is ranked with respect to the prototype or class-model corresponding to the search term

$$H = \underset{j}{\text{sort}}(P(X_j|C)) \quad \text{where } j \in \{1, N_{\text{examples}}\} \tag{2}$$

Retrieval is usually performed on a large collection of instances, and only the top of the sorted list, representing the best ranking instances, is considered as interesting. Under such a condition, a large number of classes and a massive data collection can pose a problem, since for each query there is a large number of distractors, i.e., concerning instances from all classes, other than the target class.

This becomes apparent in retrieval engines for handwritten words in historical collections [3]. In the *Monk* system, twenty books of ≈ 1000 pages each contain millions of word zones or word candidates, and the lexicon is in the order of tens of thousand word class models. From the tradition of handwriting-recognition research, it seems reasonable to start with the classification problem (Eq. (1)), using good shape features and a powerful classifier, such as, e.g., hidden-Markov models [4,5] or the support-vector machine [6,7]. For a word-mining task, such a classifier may be trained to discriminate a particular word class, and a ranked word list may be constructed, e.g., using the signed SVM discriminant value d_{SVM} for sorting. The basic assumption then is that the distance from the margin, i.e., from the instances in the distractor classes, will be a good criterion for constructing a ranked hit list for a target class. However, upon applying this approach, we observed an interesting phenomenon in the resulting hit lists. As an example, Fig. 1 shows the top-25 instances in a hit list for the word ‘Zwolle’. The performance for the word classifier on the entire training set was 100% accuracy, with a 97% accuracy on an independent test set ($k=7$ folds, $\sigma = \pm 1\%$). Following regular testing procedures for SVMs, the training and the test sets were of similar size, each containing a quarter of positive examples (typically 50) and three quarters of negative or distractor examples. However, the resulting hit list contains a number of counter-intuitive samples (e.g., speckle images) in the early ranks, followed by a strand of correct classifications which is followed by a transitional stage of occasional errors.

The impression that a problem exists is confirmed by a larger-scale analysis of the results (Table 1), also using a realistic large set containing $\approx 12 \times 10^3$ distracting word instances in the test set. The results for *accuracy* and *recall* on the realistic data set confirm

* Corresponding author. Tel.: +31 506366502.
E-mail addresses: J.P.van.Oosten@ai.rug.nl (J.-P. van Oosten), L.Schomaker@ai.rug.nl (L. Schomaker).

0031-3203/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.patcog.2013.09.006

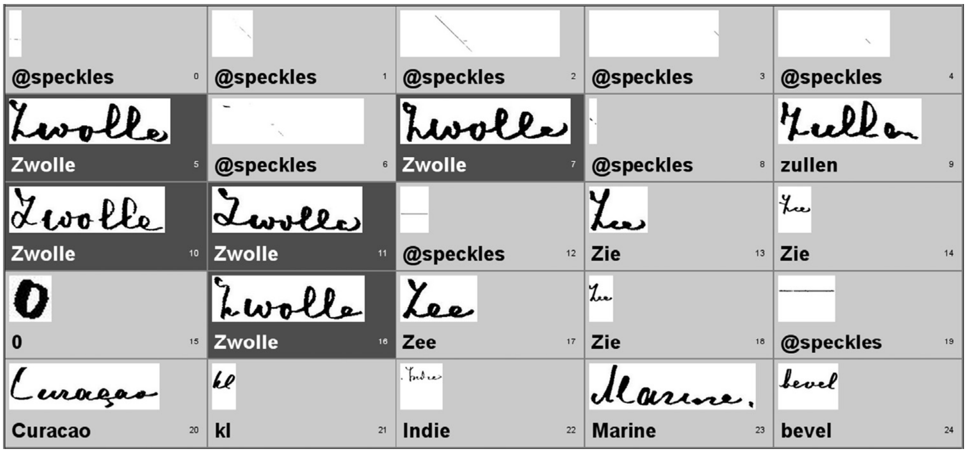

Fig. 1. First 25 instances in a hit list of the word ‘Zwolle’. Original test set performance: accuracy: 99.2%, precision: 97.6% and recall: 97.6%. Note the faulty instances in the top ranks, upper row. In a realistic test condition with 12k distractors, actual precision is as low as 2.8%.

Table 1
Counter-intuitive, low precision results for good classifiers.

Set	N_{examples}	Accuracy		Recall		Precision	
		Mean	σ	Mean	σ	Mean	σ
Test	120+	0.98	0.02	0.97	0.05	0.96	0.07
	60–120	0.97	0.03	0.95	0.10	0.91	0.13
	35–60	0.97	0.04	0.93	0.15	0.85	0.19
	7–35	0.96	0.04	0.68	0.42	0.57	0.40
+12K distractors	120+	0.99	0.01	0.97	0.05	0.26	0.26
	60–120	0.98	0.02	0.95	0.10	0.06	0.12
	35–60	0.97	0.02	0.93	0.15	0.03	0.06
	7–35	0.97	0.04	0.68	0.42	**0.01**	0.05

the hopeful expectancies which were raised by the regular training and test sets. However, the *precision* of the output drops abysmally, to about 1% in the worst cases, notably for the classes with a limited number of training examples (Table 1, lower right). It should also be noted that a number of 12K distractors (1/1200) are much more realistic than a 1/4 rule which is commonly accepted in academic testing.

It is clear that something is needed to improve on the performance. User appreciation of hit lists is of paramount importance in live and continuously trainable systems that rely on user annotation over the internet, such as Monk [3,8]. Fig. 2 shows how hit lists are used in the Monk system. Upon giving the first handful of (bootstrap) examples, a usable machine-learning system should be able to produce an acceptable ranking such that newly found instances of the same class can be easily labelled. The above, concrete observation thus gives rise to a more fundamental question: how is it possible that accuracy is not a good predictor of precision in a retrieval context?

In this study, we will (1) analyse the reason for unexpected, low precision in presumably well-performing classifiers; (2) explore a number of methods to counteract the precision drop and (3) present a convenient approach using nearest-centroid matching, with results in a similar ballpark as the abovementioned SVM approach, at the same time however, avoiding expensive training on the tens of thousands of word classes.

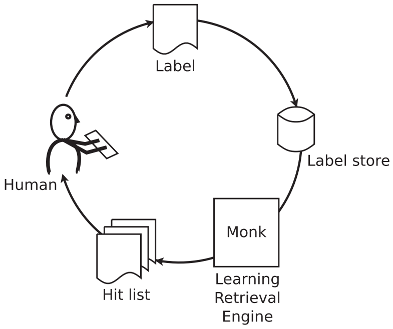

Fig. 2. Schematic overview of how users utilise the hit lists to label new word images in a continuously learning retrieval engine (Monk). A hit list is presented to the user, who produces a label for an unlabelled word. This label is stored in the label store, which is then processed by the retrieval engine to produce a new hit list. The interface facilitates the quick labelling of a large number of instances that match the query word.

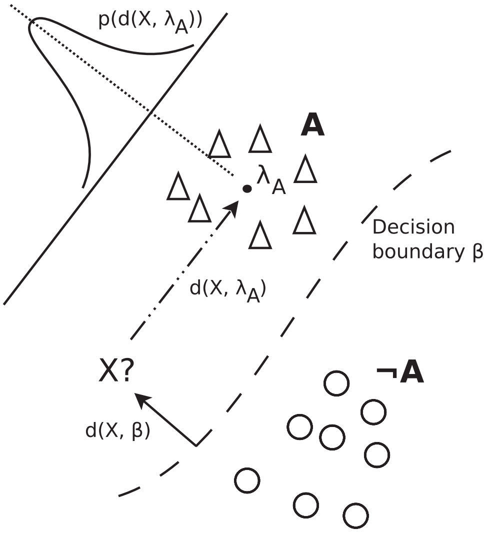

Fig. 3. Separability vs. Prototypicality: For an unknown instance X, a large distance $d(X,\beta)$ from a margin β does not imply a short distance, $d(X,\lambda_A)$ from the prototype λ_A.

2. Separability versus prototypicality

Problem: The SVM is a discriminative classifier, optimised for *classification* (Eq. (1)). The class of an unknown sample X (Fig. 3) is decided by determining on which side of the decision boundary β the sample falls. For *retrieval* purposes, it appears reasonable to use the distance to the boundary, $d(X,\beta)$, as a ranking measure: the