Vision, Video and Graphics (2005)
E. Trucco, M. Chantler (Editors)

Image classification using compression distance

Yuxuan Lan and Richard Harvey

School of Computing Sciences, University of East Anglia, U.K.

Abstract

The normalised compression distance measures the mutual compressibility of two signals. We show that this distance can be used for classification on real images. Furthermore, the same compressor can also operate on derived features with no further modification. We consider derived features consisting of trees indicating the containment and relative area of connected sets within the image. It had been previously postulated that such trees might be useful features, but they are too complicated for conventional classifiers. The new classifier operating on these trees produces results that are very similar to those obtained on the raw images thus allowing, for the first time, classification using the full trees.

Categories and Subject Descriptors (according to ACM CCS): I.4.8 [Scene Analysis]: Object recognition

1. Introduction

This section provides a very brief introduction to Kolmogorov complexity and the proposed normalised compression distance. We then describe the options for approximating this distance and show how it may be combined into a classification scheme. The remainder of the paper applies these ideas to several image classification problems and shows how the same classifier can, via the normalised compression distance, handle quite different objects.

If x is a binary string of finite length $l(x)$ and $\mathcal{U}$ is a universal machine with output $\mathcal{U}(p)$ due to program p then the Kolmogorov complexity [CT91] may be defined as,

$$K_{\mathcal{U}}(x) = \min_{p:\mathcal{U}(p)=x} l(p) \tag{1}$$

Thus $K_{\mathcal{U}}(x)$ is the shortest program that can reproduce x without error which is often written, without the subscript, as $K(x)$. We can also write the conditional Kolmogorov complexity $K(x|y)$ to mean the shortest program that can reproduce x when the program is augmented with the data y[†]. In [LCL*03] it is shown that the algorithmic information distance

$$E_1(x,y) = \max\{K(y|x), K(x|y)\} \tag{2}$$

is, up to an additive logarithmic term, the length of the shortest program to compute x from y and y from x. Furthermore (2) is known to be a metric [LCL*03]. A slight modification is to normalise (2)

$$d(x,y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}}, \tag{3}$$

to give the normalised information metric between two objects which has $0 \leq d(x,y) \leq 1$. There are several practical objections to (3). Firstly, a well-known consequence of the halting problem is that the Kolmogorov complexity is non-computable. Secondly, even if it were possible to compute $K(x)$ and $K(y)$ then, although the terms on the numerator are defined, it is not obvious how to compute them either. The later problem is easily resolved since it is known that $K(y|x) \approx K(xy) - K(x)$ where $K(xy)$ is the complexity of the joint object. The first problem is fundamental so in the paper of Cilibrasi et al. [CV04] there is an audacious step – approximate the non-computable Kolmogorov complexity, $K(x)$, with something computable such as the length, $C(x)$, of the output from a practical lossless compressor. In this case, the normalised compression distance is defined as:

$$d(x,y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}. \tag{4}$$

In Cilibrasi et al. [CV04] it is shown how this distance metric can be used to cluster highly diverse strings such as those derived from music, phylogeny trees, human languages and genetic sequences. The normalised compression distance,

[†] See Bennett et al. [BGL*98], for example, for a detailed discussion of terms.

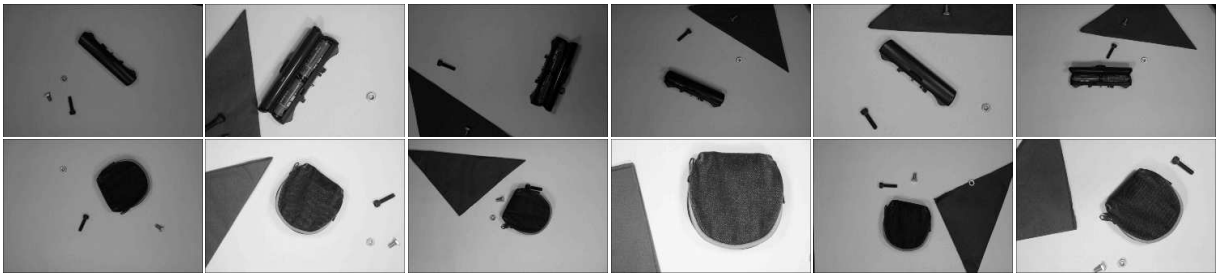

Figure 1: *Some images in the first image set showing either a battery-case or a purse on a white background. Variation in the scene was introduced by switching on or off some of the illuminating lamps, altering the location and rotation of the target object, introducing noise objects (a blue napkin, screws and a nut) and varying the zoom and height of the camera above the scene.*

(4), does not specify a compressor, but it is known that, for lossless compressors, $C(x) = K(x) + \kappa$, where κ is unknown and depends on the data and the compressor.

An alternative is described in Benedetto et al. [BL04] where the distance is defined by

$$\hat{d}(x,y) = \frac{\hat{h}(x|y) - \hat{h}(y|y)}{\hat{h}(y|y)} + \frac{\hat{h}(y|x) - \hat{h}(x|x)}{\hat{h}(x|y)}, \qquad (5)$$

in which $\hat{h}(x|y) = (\hat{C}(xy) - \hat{x})/\|y\|$ and $\hat{C}(\cdot)$ is an estimate of the compressed file size obtained using an LZ77 compressor. In this paper we use (4), not for clustering as in [CV04], but as the distance in a k-nearest neighbor classifier (kNN) [Das91] and a support vector machine (SVM) [BGV92, CV95]. To form an effective distance matrix for an SVM we require that $d(x,y)$ approximates the scalar product of two kernels so, forming $1 - d(x,y)$, provides a similarity measure that is maximal when x and y are identical. Note that kNN and SVM are both distance-based which makes them the natural choice of classifier.

2. First results

Consider a simple two-class image recognition problem with 200 images from each class. The images are captured with a Canon EOS-1D digital camera fitted with an autofocus 28-200mm zoom lens. The camera is set to 2470×1650 pixel resolution recording in RAW mode. The camera is mounted vertically above the scene which was illuminated with up to four tungsten spotlights. All exposure and focus settings were determined automatically by the camera and there was no subsequent intensity normalisation or post-processing other than that performed by the camera. One class of images always contained a battery-case and the other a purse. Before processing, the images were converted into greyscale and downsampled by a factor of ten using bicubic interpolation. Some examples of the images are shown in Figure 1. Note that, many conventional techniques would find this challenging because of the unknown, and varying, scale of the scene.

For the clustering experiments [CVW04] and in the CompLearn toolkit [Cil] the Bzip family of compressors are used. Bzip is a block-based coder that first re-orders the data using the Burrows-Wheeler transform and then uses move-to-front coding followed by Huffman coding. The compressor has around 50 bytes overhead so for small files is rather inefficient. However the major problem is that the compressor is block-based with a default blocksize of 900kB so, for large images, there is a good chance that the full string will not fit within a data block which leads to discontinuities in $C(xy)$, $C(x)$ and $C(y)$ as x and y vary in length. A similar problem is noted in [BL04] in which the LZ77 compressor is

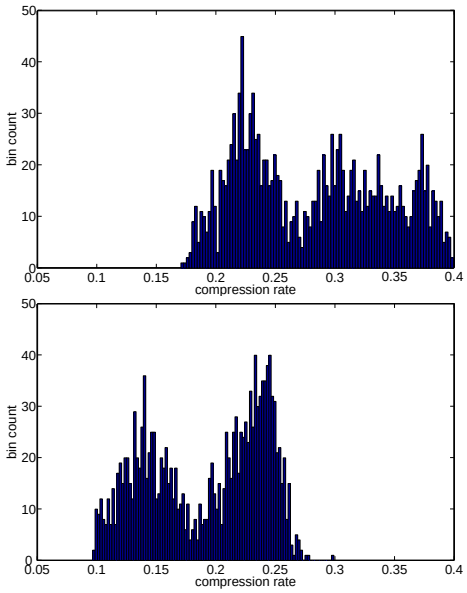

Figure 2: *The histogram of compression rate on single raw image (top) and sieve string (bottom). The mean size of the raw images is 40755 bytes, and the mean size of the sieve strings is 18602 bytes.*

used for text files. Their approach is to compute $C(xy)$ as the length of the file formed from zipping x adjoined to y. The argument for this approach is that it is an effective estimate of the relative entropy: if x is long enough then the zipper will learn x effectively. Appending string y therefore gives a good estimate of the relative entropy, provided y is not long enough to affect the learnt distributions within the zipper. Again, this method is restricted by the window size of LZ77 zipper. Furthermore, [BL04] presents no theoretical preference for (5) over (4). So here we choose (4) with $C(x)$ being approximated by the prediction by partial matching compression technique (PPM) [CW84], which, for text files, is generally acknowledged to be the best lossless compressor available. In our tests we have used implementations known as PPMZ [Blo] and, for this paper, BICOM [Tim]. Both of which use unbounded PPM models (the PPM* algorithm [CT97]) with local order estimation and secondary escape estimation. Each image is stored as a string of unsigned eight-bit integers. There are small differences in file size that depend on how the image is rasterised so we take care to rasterise all images in the same way (vertical scans in this paper). There are also several possibilities for combining files, here we concatenate files since this works best with PPM* - alternatives include interleaving pixel by pixel or in blocks. The top of Figure 2 shows the distribution of the fractional size of the compressed file compared to the original size – the mode of the distribution is around 0.23 bits per bit.

Using leave-one-out testing and a nearest neighbor classifier ($k = 1$) the error rate is 0.105. Leave-one-out testing was simulated using a model classifier that picked the classes randomly ($p = 1/2$). Over 100,000 trials the mean error was 0.500 with standard deviation 0.0251 and minimum/maximum error of 0.395/0.610 which implies that we may confidently reject the hypothesis that this result arose by chance. $k = 1$ provides the best performance which suggests that a larger dataset is desirable. The support vector machine we applied is a public domain toolbox [Caw00], in which the SMO algorithm [Pla98] is used for optimisation, and the regularisation parameter C, is selected by a grid search using five-fold cross-validation. The error rate by SVM was 0.225 with all the data as support vectors.

3. Discussion of first results

Note that the method completely avoids any feature extraction and measures only the compressibility of pairs of images compared to the size of the images when compressed individually. A natural alarm might be that the two classes were actually distinguishable using some trivial features, such as intensity. Figure 3 shows a scattergram of the mean intensity and its variance for the two classes and also the combined intensity histograms. There is considerable overlap between the classes. Computing a t-test for the difference between the means of the means in Figure 3 gives $t \approx 3.5$,

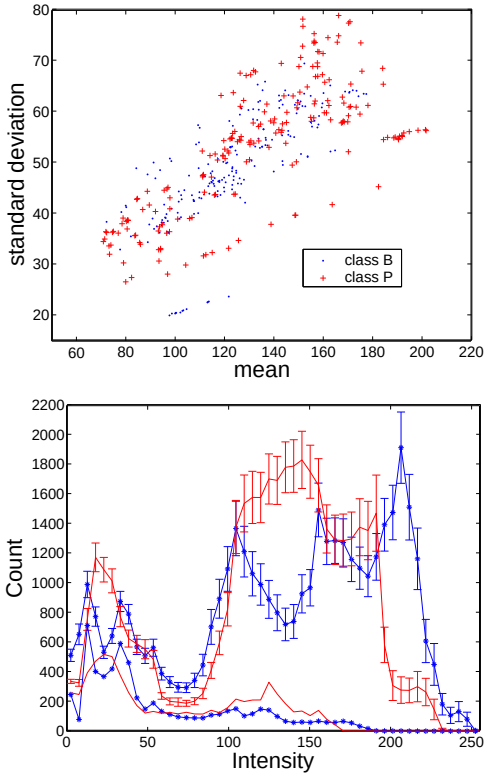

Figure 3: *Top: scattergram of the mean and standard-deviation of the intensity in each image. Bottom: intensity histograms of the two classes (50 bins). The upper two curves show the mean histogram ($\pm$ 1 standard error for the two classes. The lower two curves are the median histogram. The bin width was chosen according to Scott's rule ($b \approx 3.5\sigma N^{-1/3}$) [Sco79] where σ is the mean of the standard deviations computed from each image and N=400.*

$\nu \approx 351$ which allows us to reject the null hypothesis. In other words the classes are separable on mean intensity alone. We can therefore construct a benchmark classifier based on comparing intensity histograms. Computing a χ^2 distance between the 50 bin histogram of a particular image and the mean histogram for each class (we also computed the L1 distance but found the results indistinguishable from random guessing) allows the construction of a classifier with an error rate of 0.34. It is gratifying that the compression-based classifier out-performs an intensity-based classifier, even on a dataset that is partly separable on intensity. As confirmation, we have histogram-equalised all images to the mean histogram, in which case the χ^2-based classifier error rate increases to 0.43 (which is indistinguishable from guessing) whereas the error rate of the best performing compressed-based classifier increases to 0.115 (kNN with $k = 33$) which is still confidently better than chance. This illustrates another desirable feature of the compression-based classifiers

– they automatically select discriminating information from the training data.

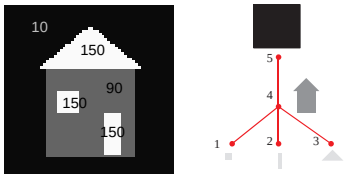

Figure 4: *Left: a simple image, showing the grey level intensity of each component. Right: the corresponding sieve tree with each node representing a granule.*

An alternative intensity normalising transform is a set of nested greyscale alternating sequential filters known as a *sieve* [BHLA96, DL00]. The algorithm has its basis in graph morphology in which $G = (V,E)$ is a graph with a set of vertices, V which label the pixels and a set of edges, E, which define the neighbourliness of pixels. The notation is flexible and handles n-dimensional images with any connectivity. The image intensities may be represented as $f(v), v \in V$. For scales, $s \geq 1$, let $\mathcal{C}_s(G)$ denote the set of connected subsets of G with s elements. Then, with $x \in V$, $\mathcal{C}_s(G,x) = \{\xi \in \mathcal{C}_s(G) \mid x \in \xi\}$. denotes the set of connected sets of s pixels that contain pixel x. This allows a compact definition of an *opening*, ψ_s, and *closing*, γ_s, of scale s. The morphological operators, $\psi_s, \gamma_s, \mathcal{M}_s, \mathcal{N}_s : Z^{\rightarrow} Z^V$, may be defined for each integer, $s \geq 1$, as

$$\psi_s f(x) = \max_{\xi \in \mathcal{C}_s(G,x)} \min_{u \in \xi} f(u), \tag{6}$$

$$\gamma_s f(x) = \min_{\xi \in \mathcal{C}_s(G,x)} \max_{u \in \xi} f(u), \tag{7}$$

and

$$\mathcal{M}_s = \gamma_s \psi_s, \qquad \mathcal{N}_s = \psi_s \gamma_s. \tag{8}$$

Thus $\mathcal{M}_s$ is an opening followed by a closing, both of size s and in any finite dimensional space. The M- and N-sieves of a function, $f \in Z^V$ are defined in [BHLA96] as sequences $(f_s)_{s=1}^{\infty}$ with the M- and N-sieves being:

$$f_1 = \mathcal{M}_1 f = f, \text{ and } f_{s+1} = \mathcal{M}_{s+1} f_s \tag{9}$$

$$f_1 = \mathcal{N}_1 f = f, \text{ and } f_{s+1} = \mathcal{N}_{s+1} f_s \tag{10}$$

for integers, $s \geq 1$. Note that these M- and N-sieves are alternating sequential filters that do not use structuring elements but merge connected sets instead. The differences between successive outputs

$$d^s = f_s - f_{s-1} \tag{11}$$

are called *granule functions* and non-zero connected regions within d^s are called *granules* denoted by d_j^s where $j = 1 \ldots N_G(s)$ indexes the number of granules, $N_G(s)$, at scale s. As scale s increases, $N_G(s)$ decreases, since the granules are larger. At the final scale there is only one granule

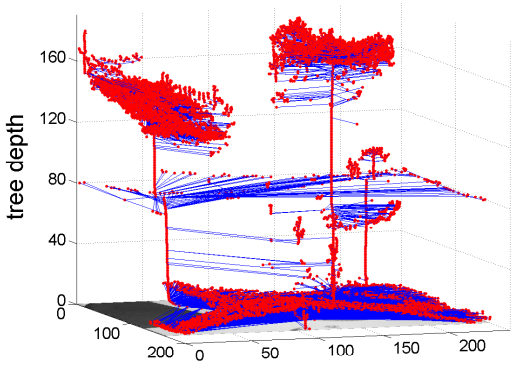

Figure 5: *Sieve tree of the last image in Figure 1(*247×165 *pixels), which consists of 11465 nodes in 191 levels.*

that is the size of the image. The granules may be recombined through addition. Thus the processor forms a lossless transform.

A scale tree, $T = (N,A)$ may be built using the output of a sieve $(d_s)_{s=1}^{S}$ and is also a graph with a set of vertices, or nodes, N, and edges, A. If the image contains S pixels then the root of the tree, $\mathcal{R}(T)$ maps to d_1^S which is the whole image. If $a \in A$ with $a = (n_p, n_c)$ then n_c is a child of n_p and $d_{n_c}^{s_c} \subset d_{n_p}^{s_p}$. Figure 4 shows an example tree in which the root of the tree represents the whole image, nodes are connected sets and tree-edges indicate containment. In other words because the sieve is removing local extrema, granules at some scale s_c are always contained within granules at some greater scale, s_p, unless $s_c = S$ in which case it is the root. Since the system operates on maxima and minima only, the shape of the connected sets and the order in which they are removed is invariant under any intensity transformation which preserves the relative ordering of the image greyscale.

For the purpose of object recognition, a similarity measurement of two trees is desirable. However, in practice, the trees can become very complex, as shown in Figure 5, which the conventional tree comparison methods, e.g. tree isomorphism and tree-to-tree correction [CFSV04], would struggle to handle. An alternative is to convert the tree hierarchy to string, which is a common and simple format for tree processing. Then our compression-based measurement can be applied for pair of tree strings in terms of similarity distance of these trees. A depth-first method is adopted here which tacitly makes the relationship between parent and child be more important than the relationship between brothers. For Figure 4, the traversal path will be [5 4 3 0 2 0 1], where 0 indicates moving up a level. The string therefore is a variable length data structure.

Currently we store only area with node. If the areas of granules 1 to 5 are 60, 90, 294, 1318 and 90000 respectively, a string representation could be [90000 1318 294 0

Figure 6: *Some images in the complex image set. The class objects are unchanged, a purse and a battery case, but there are now many more noise objects, complicated backgrounds such as magazines and newspapers, and a greater variety of viewpoints and lighting variations. In some cases the class objects are occluded.*

90 0 60]. An alternative is to store the ratio of the area of each node to its parent instead. In this case, the string would be [90000 0.0146 0.2231 00.0683 0 0.0486]. It is this notation which we prefer since it removes the scale variation due to zoom and location of the camera[‡]. The first element in the string encodes the size of the root, which is the number of pixels in the image, this is ignored since it is usually non-informational. As a final step, the ratios in the string are quantised using a max-entropic quantisation scheme into 1000 bins (and 5000 bins for the complex image set in the later experiment). The final result is written out as a string of unsigned 16 bit integers.

4. New results

A more complicated image dataset (1492 images in total), containing the simple images as a subset, is also used to test our technique. Figure 6 shows some of them. The scale variation remains but there are additional variations which make the problem substantially more challenging.

The classifiers were tested using ten-fold cross-validation which has the advantage over leave-out-one testing of allowing the computation of standard error. Figure 7 shows the results from both the first image set and the more complex case just described. A simple simulation of a classifier guessing at random using the same cross-fold validation technique gave, in 100,000 trials a mean error of 0.5000 with a standard deviation of 0.0130, the smallest error was 0.4450 which implies that we may confidently reject the hypothesis that any of the results arose by chance. The *k*NN for sieve strings now indicates the characteristic shallow U-shape giving evidence that the strings now have a neigbourhood of similarity.

‡ The ratio of areas of planar objects are invariant under affine transformations which also makes their use invariant.

McNemar's test [GC89, Die98] is used to determine if the difference in the errors of a pair of classifiers is significant. The test requires the construction of the joint performance of classifiers, as shown in Table 1, which indicates

		B	
		Correct	Incorrect
A	Correct	N_{00}	N_{01}
	Incorrect	N_{10}	N_{11}

Table 1: *Joint performance of classifier A and B on two-class problem*

the agreement(N_{00}, N_{11}) and disagreement(N_{01}, N_{10}) of two classifiers when parsing the same data set. Only the disagreement is used in McNemar's test for it contains the information of the performance difference. Assuming that A and B are not significantly different, if only one of them misclassifies on a pattern, it is equally likely to be A or B. Therefore, for the null hypothesis H_0 (that A and B are not significantly different), N_{01} and N_{10} obey the binomial distribution $\mathcal{B}(k,q)$ in which $k = N_{10} + N_{01}$, $q = 1/2$. The p value is computed using the two-tailed test:

$$p = \begin{cases} 2\sum_{m=N_{10}}^{k} \binom{k}{m} \left(\frac{1}{2}\right)^k & N_{10} > k/2 \\ 2\sum_{m=0}^{N_{10}} \binom{k}{m} \left(\frac{1}{2}\right)^k & N_{10} < k/2 \end{cases} \tag{12}$$

H_0 will be rejected if the p-value is smaller than a given significance level α. For repeated tests, there are arguments for and against a Bonferroni adjustment of α [Per98, Zal97] so here we report p-values. According to Figure 8, for the classifiers on simple images, the only two classifiers which are not significantly different are the *k*NNs, the p-values for all the others approximate 0 (smaller than 5×10^{-5}), showing that similarity of these classifiers is unlikely. For classifiers

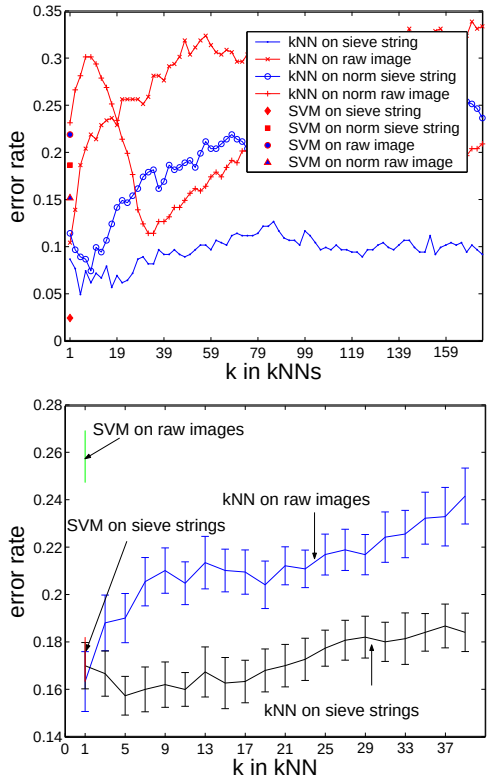

Figure 7: *Top: testing error on the simple image set, using leave-one-out cross-validation. Bottom: testing error on the complex image set using ten-fold cross validation. For the simple set we also show the effect of intensity* normalisation *(denoted 'norm' in the key) via histogram equalisation. The x-axis shows k for the kNN. The* raw *image is a raster-scanned version of the image.*

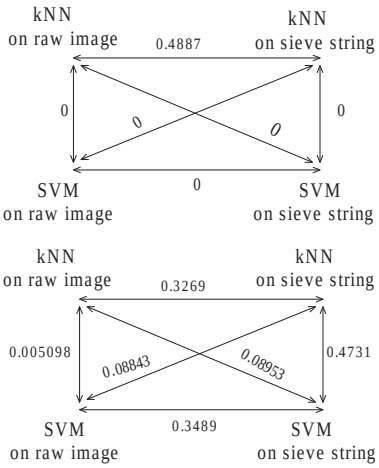

Figure 8: *p-value for pairwise comparison of classifiers. Top: classifiers for simple images. Bottom: classifiers for complex images. In all cases k is that which gives the lowest test error.*

on complex images, no pair of classifiers can reject the null hyperthesis at a significance of 0.01 which means that, while the results on bottom of Figure 7 are promising, they would benefit from more data.

Figure 9 shows a further interesting interpretation of these results showing the nearest neighbour match to a particular image string from the set of all the other 1491 images. The images are numbered in the order in which they were taken so the diagonal line indicates that the compression-based nearest neighbour is usually the most visually similar one. Exceptions are, for example, images from 500 to 742, which are characterised by all having similar backgrounds. Note that the sieve string appears to be much more robust than the raw image string, as there are fewer points in the off-diagnal squares.

5. Ground truth data

In these experiments, the segmentation problem has been ignored. Yet, as mentioned previously the background introduced some bias which affects the performance of the classifier. This implies the potential of better performance by introducing a segmentation. The foreground of a subset of images (204 in total) taken from the complex images, was hand-labelled to produce the ground truth. For each image tree T with nodes $A_1 \ldots A_{|T|}$, the optimal node A_i which best covers the foreground segment, M, is the one that minimize the normalizes XOR error $E_\oplus$:

$$E_\oplus = \frac{|A_i \oplus M|}{|A_i| + |M|}, \quad i = 1 \cdots |T| \tag{13}$$

The subtree rooted at A_i represents the foreground object (as in Figure 10), and the substring corresponding to it can be chopped from the sieve string. Besides the *k*NN and the SVM, histogram classifiers which compute the χ^2 distance between alphabet histogram of each pattern and the mean histogram of each class are also constructed, with bin size optimized following Scott's rule [Sco79]). In Figure 11, the substring classifiers work better than those that use whole string, indicating the positive effect of segmentation for learning. Of course the segmentation problem remains but it is now a one-dimensional problem because strings constructed by depth-first parsings of sieve trees have the property that connected sets in the image aways map to contiguous sub-strings. Figure 12 shows the nearest neighbours of 104 segmented sieve strings. The *x*-axis is the index of the test images which here have complex backgrounds. The *y*-axis is the nearest match from the set with simple background. The indices were chosen so that 1 to 52 are class 1 and re-

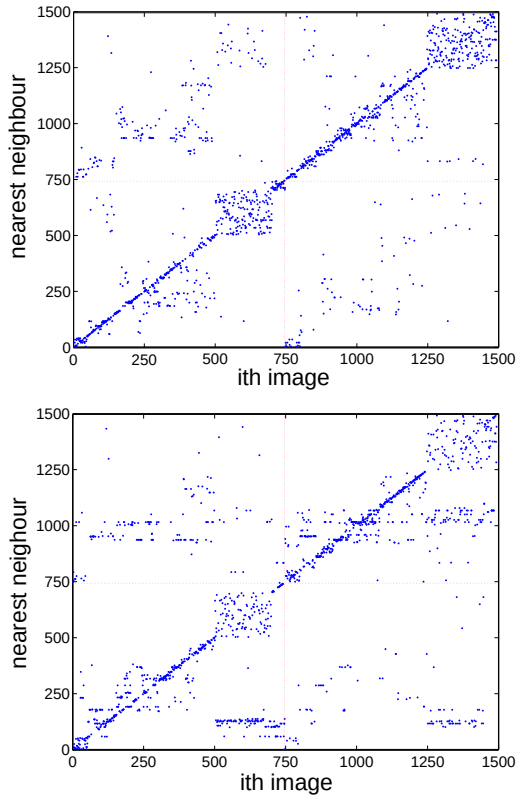

Figure 9: *The nearest neigbour of each image from the complex image, using sieve string distance (top) or raw image string distance (bottom). The first 742 images are from class 1 and the rest from class 2. The red dashed line which sub-divides the space into four squares marks the separating point of two classes. Points that fall into first and third squares are positive matches.*

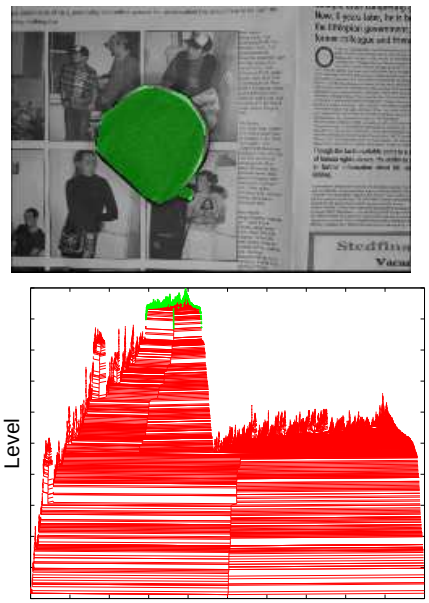

Figure 10: *One image from the experiment. Top: the hand-segmented foreground (the purse). Bottom: the sieve tree of the image, in which y axis stands for the depth of node. Highlighted nodes (shown green if printed in colour or light grey in greyscale) make up the subtree that describes foreground object.*

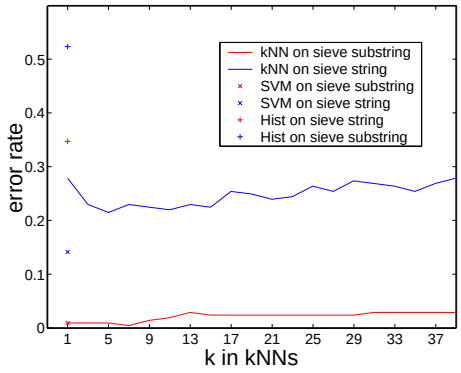

Figure 11: *The performance of classifiers based on whole sieve strings and sieve substrings, evaluated by leave-one-out cross validation. 'Hist' stands for the histogram method.*

mainder are class 2. We infer from this result that there is enough discriminating structure in the substring that represents the object to preserve the low error rate.

6. Conclusion

This paper has applied the normalised compression distance in a classification framework using the prediction by partial matching compressor. Although the method has its origins in Kolmogorov complexity, the resulting classifier is simple to implement and takes advantage of any developments in compression – an area where there are considerable rewards for improved algorithms. A great strength of the method is that it does not require an explicit feature extraction step which has been demonstrated by applying identical classifiers to two types of data that differ greatly: the raw image and the output of a sieve tree. Furthermore the method covers variable-length features without any special engineering. We find that, even though the sieve tree string is considerably smaller than the original image, the classification performance is, on the whole, better with sieve trees than without. This confirms and extends the results of, for example, [DL00] in which it was shown that matching these trees might be viable technique for image retrieval.

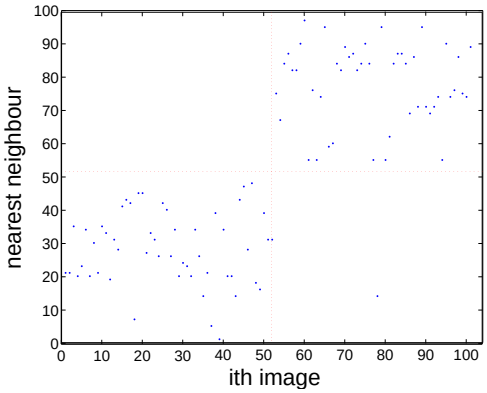

Figure 12: *The nearest neighbour scatter graph. The 1nn classifier is trained and tested with images of different background groups. The error rate is below* 1%

References

[BGL*98] BENNETT C. H., GÁCS P., LI M., VITÁNYI P. M. B., ZUREK W. H.: Information distance. *IEEE Transactions on Information Theory 44*, 4 (July 1998), 1407–1423.

[BGV92] BOSER B., GUYON I., VAPNIK V. N.: A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on computational learning theory* (Pittsburgh, 1992), ACM, pp. 144–152.

[BHLA96] BANGHAM J., HARVEY R., LING P., ALDRIDGE R.: Scale-space preserving transforms in many dimensions. *Journal of Electronic Imaging 5*, 3 (July 1996), 282–299.

[BL04] BARONCHELLI A., LORETO V.: Language trees and zipping. http://arxiv.org/abs/cond-mat/0403233, March 2004.

[Blo] BLOOM C.: PPMZ Toolkit. http://www.cbloom.com/src/ppmz.html.

[Caw00] CAWLEY G. C.: MATLAB support vector machine toolbox (v0.55β), 2000. http://theoval.sys.uea.ac.uk/svm/toolbox.

[CFSV04] CONTE D., FOGGIA P., SANSONE C., VENTO M.: Thirty years of graph matching in pattern recognition. *International Journal of Pattern Recognition and Artificial Intelligence 18*, 3 (2004), 265–298.

[Cil] CILIBRASI R.: Complearn toolkit. http://complearn.sourceforge.net/.

[CT91] COVER T. M., THOMAS J. A.: *Elements of Information Theory*. Wiley, 1991.

[CT97] CLEARY J. G., TEAHAN W. J.: Unbounded length contexts for PPM. *The Computer Journal 40*, 2/3 (1997), 67–75.

[CV95] CORTES C., VAPNIK V.: Support vector networks. *Machine Learning 20* (1995), 1–25.

[CV04] CILIBRASI R., VITÁNYI P. M.: Clustering by compression. http://arxiv.org/abs/cs.CV/0312044, April 2004.

[CVW04] CILIBRASI R., VITÁNYI P., WOLF R. D.: Algorithmic clustering of music based on string compression. *Computer Music Journal 28*, 4 (winter 2004), 49–67.

[CW84] CLEARY J., WITTEN I.: Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communication COM-32*, 4 (April 1984), 396–402.

[Das91] DASARATHY B. V. (Ed.): *Nearest neighbour (NN) norms: NN pattern classification techniques*. IEEE Computer Society, Washingtion, DC, 1991.

[Die98] DIETTERICH T. G.: Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation 10*, 7 (1998), 1895–1924.

[DL00] DUPPLAW D., LEWIS P. H.: Content-based image retrieval with scale-spaced object trees. In *Proceedings of SPIE: Storage and Retrieval for Media Databases 2000* (2000), Yeung M. M., Yeo B.-L.„ Bouman C. A., (Eds.), vol. 3972, pp. 253–261.

[GC89] GILLICK L., COX S.: Some statistical issues in the comparison of speech recognition algorithms. In *Proceedings, ICASSP* (1989), vol. 1, pp. 532–535.

[LCL*03] LI M., CHEN X., LI X., MA B., VITÁNYI P.: The similarity metric. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms* (2003), Society for Industrial and Applied Mathematics, pp. 863–872.

[Per98] PERNEGER T. V.: What's wrong with Bonferroni adjustments. *British Medical Journal 316* (April 1998), 1236–1238.

[Pla98] PLATT J.: Fast training of support vector machines using sequential minimal optimization. In *Advances in kernel methods - support vector learning*, Scholkopf B., Burges C.„ Smola A., (Eds.). MIT Press, 1998.

[Sco79] SCOTT D. W.: On optimal and data-based histograms. *Biometrika 66*, 3 (December 1979), 605–610.

[Tim] TIMMERMANS M.: BICOM. http://www3.sympatico.ca/mt0000/bicom/.

[Zal97] ZALZBERG S. L.: On comparing classifiers: pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery 1*, 3 (1997), 317–327.