Mathematical Symbol Recognition with Support Vector Machines

Christopher Malon [a,*], Seiichi Uchida [b], Masakazu Suzuki [a]

[a] *Faculty of Mathematics, Kyushu University*

Hakozaki 6–10–1, Higashi-ku, Fukuoka, 812–8581 Japan

[b] *Faculty of Information Science and Electrical Engineering, Kyushu University*

Motooka 744, Nishi-ku, Fukuoka, 819–0395 Japan

Abstract

Single–character recognition of mathematical symbols poses challenges from its two-dimensional pattern, the variety of similar symbols that must be recognized distinctly, the imbalance and paucity of training data available, and the impossibility of final verification through spell check. We investigate the use of support vector machines to improve the classification of InftyReader, a free system for the OCR of mathematical documents. First, we compare the performance of SVM kernels and feature definitions on pairs of letters that InftyReader usually confuses. Second, we describe a successful approach to multi–class classification with SVM, utilizing the ranking of alternatives within InftyReader's confusion clusters. The inclusion of our technique in InftyReader reduces its misrecognition rate by 41%.

Key words: Support vector machine; OCR; Mathematical document

* Corresponding author. Tel: +81 92 642 7047; Fax: +81 92 642 2789.
Email addresses: `malon@math.kyushu-u.ac.jp` (Christopher Malon),

1 Introduction

Mathematics is the universal language of scientific literature, but a computer may find it easier to read the human language in which surrounding text is written. The failure of conventional OCR systems to treat mathematics has several consequences:

- Readers of mathematical documents cannot automatically search for earlier occurences of a variable or operator, in tracing the notation and definitions used by a journal article.
- The appearance of mathematics on the same line as text often confounds OCR treatment of surrounding words.
- Equations can only be represented as graphics by semantic transformation systems, such as those converting digital documents into braille for accessibility by blind readers [17].

Mathematical OCR was investigated as early as 1968 [1]; a survey of its difficulties and previous approaches may be found in [2]. A modern system competing with ours has achieved a 93.77% single-character recognition rate [9]. OCR of mathematics differs markedly from typical text recognition because its single-character recognition phase must be followed by a *structural analysis* phase, in which symbol relationships involving superscripts, subscripts, fractions, and matrices must be recovered. The two-dimensional arrangement affects not only structural analysis but single-character recognition itself, because typical assumptions about bounding boxes and baselines are violated.

`uchida@is.kyushu-u.ac.jp` (Seiichi Uchida), `suzuki@math.kyushu-u.ac.jp` (Masakazu Suzuki).

Even in relatively simple equations such as

$$\phi|_{\mathbb{C}}(z) = \exp(zN_\phi)$$

the subscript-positioned capital blackboard bold $\mathbb{C}$, whose base is nearly aligned with that of the vertical bar, might be mistaken for a lower-case letter.

In this paper, we focus on the single-character recognition phase that precedes structural analysis. We have addressed structural analysis in [8], [15], [22], and [19]. The single-character OCR of mathematics poses challenges that, if not unique, place it alongside the most difficult human languages to recognize. The recognition problem consists of about 1,000 classes, many with little existing ground truth data. Certain distinct letters, such as Latin v and Greek ν, are in close resemblance. Most unusually, we desire the distinction of *styles.*

In typical mathematical usage, different styles of the same letters will have completely different meanings. The problem is most severe not in engineering, but in pure mathematics. For example, within a single article in p-adic representation theory, the bold letter $\mathbf{G}$ often will represent a group over an algebraically closed field, the plain italic G will represent its rational points over a p-adic field k, and sans-serif G a reductive quotient over the residual field $\overline{k}$, with German $\mathfrak{g}$ used for a Lie algebra. Calligraphic $\mathcal{A}$ may represent a simplicial complex, and italic A a torus. (See, *e.g.*, [6].) An optical character recognizer that does not keep these letters distinct would be practically useless in this branch of algebra. However, within a single style, *fonts* (Computer Modern, Times, Helvetica, *etc.*) should not be distinguished, so that mathematical formulas can be compared between articles, regardless of the fonts the publisher has chosen.

OCR problems were considered very early in the development of SVM, with promising results. An experiment by Cortes and Vapnik [5] achieved 95.8% accuracy on handwritten digits in the US Postal Service database. More particularly, in character recognition of human languages with hundreds of distinct characters, SVM have achieved promising results, for example, in handwritten Chinese (99.0%, [7]) and printed Ethiopian (91.5%, [14]). Recently, SVM has been applied to handwritten mathematics on a blackboard [21], but to our knowledge, OCR of printed mathematics using SVM has not been investigated before.

This paper describes an experiment using SVM to improve multi-class classification by an existing OCR system. This OCR system is a purified version of the InftyReader, a freely available OCR engine for mathematics, described in [19]. First, we study the ability of various kinds of SVM, as binary classifiers, to distinguish pairs of letters that confuse InftyReader. Then, we show how the classifiers may be integrated with the system to improve its multi-class classification ability.

2 Ground truth data

The InftyProject defined a set of 1,629 mathematical characters to be distinguished, and released several databases of ground truth, containing both single-character and structural recognition results, starting with InftyCDB-1 [20], whose composition is described in [23]. Because some mathematical symbols occur very rarely, it is necessary to choose between extracting each symbol from documents in their entirety, or seeking out samples of particularly rare characters to provide more uniform representation. The newest databases of the InftyProject, InftyCDB-3-A and InftyCDB-3-B [18], targeted at single-character recognition experiments, cover both approaches. InftyCDB-3-B represents twenty articles from advanced mathematics journals at full length; it consists of a tenth of the samples of InftyCDB-1, chosen by clustering techniques. InftyCDB-3-A [16] aims to represent rare characters by more samples; it includes not only journal articles, but font samples, and multiple scans of letters at different greyscale thresholds. We use InftyCDB-3-A (188,752 characters, representing 384 symbol entities from 326 documents) for training, and InftyCDB-3-B (70,637 characters, representing 275 symbol entities from 20 documents) for testing. No database includes samples of all 1,629 symbol entities defined by the Infty Project.

In InftyCDB-3-A and InftyCDB-3-B, a sample of ground truth data for a symbol entity consists of a black and white bitmap image of that symbol in isolation, framed inside its bounding box, extracted from a scanned physical document. Thus, the data set does not test the OCR system's ability to group nearby components together, as in the two parts of the symbol '$\leq$.' Because *spatial context* is lost, some pairs of symbols, such as hyphens and underscores, must be regarded as the same. InftyReader distinguishes among these characters after an entire line of text or mathematical expression was read. Also, light or dark printing can affect whether a character should be regarded as bold or not; InftyReader makes such decisions after the density of all characters on the page is known. Thus, bold characters are thus identified with their non-bold counterparts. German letters, which number too few, and touching and broken characters, are excluded from our training and testing data. [1]

[1] In the earlier ground truth database InftyCDB-1, touching characters comprise 1.25% of the character samples. We address the segmentation of touching characters

In Table 1, we present representatives of the 384 symbol entities appearing in InftyCDB-3-A. Figure 1 shows the number of training samples available for each of these classes. Although an average class has between 500 and 1,000 training samples, more than 75 classes have fewer than 50 training samples.

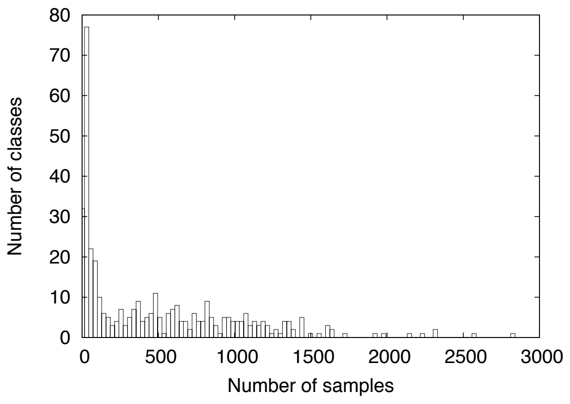


Fig. 1. Histogram of number of training samples, by class.

3 Confusion Matrix

The engine of InftyReader typically makes use of contextual information, but for this experiment, we distill it to ignore information about a character's size or surrounding characters. The purified engine simply classifies images framed inside bounding boxes. By running the purified InftyReader engine on the training data, we produce an integer-valued confusion matrix, with rows that count ground truth and columns that count recognition results. Every nonzero off-diagonal entry of this matrix represents a *confusing pair*, for which an SVM should be trained. There are 771 confusing pairs, counted as unordered pairs.

in [13], with a strategy that only needs to query SVM that are trained to recognize correctly segmented input images.

In the confusion matrix, each row represents Infty's recognition result and each column represents ground truth. The set of nonzero entries from each row of the confusion matrix represents a *confusion cluster*. The sizes of these clusters are indicated in Figure 2. Most clusters consist of fewer than five alternatives, and the biggest cluster contains 26 alternatives. As the figure shows, the confusion matrix is relatively sparse, and performing multi-class classification only on confusing alternatives, instead of all 384 symbols, significantly reduces complexity. Each cluster can be partially ordered by the likelihoods of each alternative, as indicated by the values of the corresponding matrix entries. This ordering will be utilized in our multi-class classification strategy later.

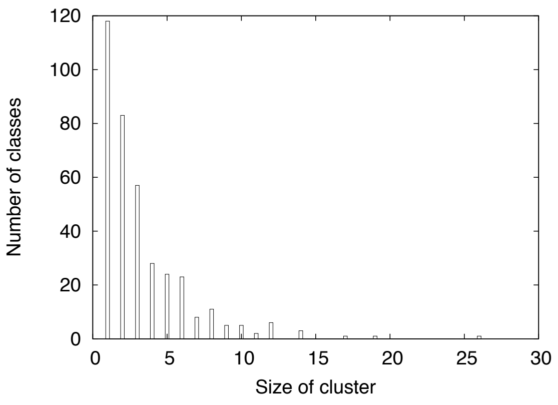


Fig. 2. Sizes of confusion clusters.

4 Pairwise classification with SVM

4.1 Features for SVM training

Some of our SVM are trained with directional histograms of the contour, introduced by Kimura *et al* [12] for Japanese handwriting recognition. For a single mesh, these feature vectors are constructed from directional histograms, measuring the amount of horizontal, vertical, and diagonal contour, in each position of the mesh. A single pixel contributes to the mesh position in which it lies, and possibly to several neighboring positions, as determined by mask functions over the bitmap. These mask functions sum to one everywhere in the bitmap.

The recognition engine for InftyReader uses differently sized meshes, in case a character is especially tall or short. Depending on a character's aspect ratio, data from a 3×5, 5×5, or 5×3 mesh may be more significant. Our 221–dimensional "directional" feature vectors consist of the aspect ratio, and three blocks, representing the directional contour histograms from the three mesh sizes. Depending on the character's aspect ratio, data from only one or two of the blocks will be regarded as significant and used for training or testing. For example, an input 'l' that has aspect ratio exceeding 1.7 will use only the 5×3 block, and the aspect ratio, for classification. [2]

In [24], Vapnik states the philosophy that, in contrast to classical approaches that work best with "strong features," "it is not important what kind of 'weak feature' one uses; it is more important to form 'smart' linear combinations." As an extreme example of this philosophy, Cortes and Vapnik's original SVM study of the USPS handwritten digit database [5] utilizes (smoothed, cen-

[2] The SVM uses the 221–dimensional feature vector, with the insignificant blocks' coordinates changed to zero, for training and testing. The naive classifier assigns a testing sample to a class by finding the nearest centroid in the feature subspace consisting of blocks that are significant for that sample. Within a single class, not every training sample will have the same significant blocks, so the centroids for each block are computed separately.

tered, de-slanted) bitmap images as feature vectors. Bitmaps as feature vectors, sometimes processed by principal component analysis, linear discriminant analysis, or nonlinear normalization, also have been the basis of more modern OCR experiments with SVM ([7], [4], and [14]).

To investigate whether the style-but-not-font distinction aspect of our recognition problem makes bitmap-based approaches less effective, or rather if directional features discard too much potentially useful information, we train another set of SVM with bitmap-like feature data. Because characters appear in bounding boxes of different aspect ratios, we cannot use raw bitmaps directly. Rather, we impose a 20 by 20 grid onto each bitmap, and measure the blackness in each grid position. Taking these measurements together with the arctangent of the aspect ratio, we obtain 401–dimensional "density" feature vectors.

4.2 Benchmark: A naive classifier

Ideally, we would compare performance of the SVM against the pure Infty recognizer itself. However, the pure Infty recognizer does not solve a binary classification problem like the SVM classifiers do. We can only say that the rate at which it picks a class A over a class B in binary selection should be greater than the rate at which it selects A out of all possible classes, and vice versa. These two bounds typically yield an interval too wide to be informative, so we implement a naive binary classifier as a more precise benchmark.

The naive classifier is constructed by recording the centroids of the sets of feature vectors representing instances of each symbol in the training data. We use the directional feature vectors for this construction. The naive classifier can perform either multi-class or binary classification; in any case, it assigns a test sample to the class with the closest centroid.

4.3 SVM training and performance

Altogether, we consider five forms of SVM constructions. On the directional features, we construct SVM with linear, Gaussian, and cubic polynomial kernels. On the density features, we construct SVM with linear and cubic polynomial kernels. These kernels have the forms:

$$\begin{aligned} K_{linear}(\vec{x},\vec{y}) &= \vec{x}\cdot\vec{y} && (1)\\ K_{Gaussian}(\vec{x},\vec{y}) &= e^{-\gamma\|\vec{x}-\vec{y}\|^2} && (2)\\ K_{cubic}(\vec{x},\vec{y}) &= (\gamma\vec{x}\cdot\vec{y}+1)^3. && (3) \end{aligned}$$

Support vector machines are trained to perform binary classification by solving the following optimization problem. Given training data with feature vectors $\vec{x_i}$ assigned to class $y_i \in \{-1, 1\}$ for $i = 1, \ldots, l$, the support vector machines solve

$$\begin{aligned} \min_{\vec{w},b,\vec{\xi}} \quad & \frac{1}{2}K(\vec{w},\vec{w}) + C\sum_{i=1}^{l}\xi_i && (4)\\ \text{subject to } & y_i(K(\vec{w},\vec{x_i})+b) \geq 1-\xi_i \\ & \xi_i \geq 0 \end{aligned}$$

where $\vec{\xi}$ is an l–dimensional vector, and $\vec{w}$ is a vector in the same feature space as the $\vec{x_i}$ (see, *e.g.*, [10]). We use the LibSVM software [3] to train SVM classifiers.

The performance of a binary SVM classifier is evaluated according to a measure which we call the *min-recall.* This value is the smaller of its recognition rates for either class.

Before training an SVM, the soft margin C and any parameters appearing in the kernel K must be chosen in advance (here, γ). For the linear and Gaussian kernel experiments using directional features, we choose these parameters by five-fold cross validation. Each training document is assigned, in its entirety, at random to one of five sets. For each binary classification problem, the cross-validation accuracy for a choice of parameter values is computed by the leave-one-out method. Parameter choices are inspected within a grid in logarithmic space, and the grid is expanded until the accuracy stabilizes or begins decreasing at all boundaries, or until an eight-minute timeout. The parameter choice producing the highest cross-validation accuracy is used once more to train the final SVM for the problem on the entire training set. This procedure cannot be performed on a binary classification problem if all the training data for either class is concentrated in a single one of the five sets; for the 17 (of 771) pairs where we have so little data, we do not construct a binary SVM.

In fact, the parameter choice is rarely important for the linear SVM; up to the hardest soft margin considered, accuracies typically remain the same, as one would expect if the data were linearly separable. A softer hard margin produces a 3% or greater improvement in min-recall on four pairs, and the constant choice $C = .01$ produces the best accuracies on training data overall. For the Gaussian kernel, as well, there is a parameter setting that yields cross-validation accuracies on each problem that are nearly as high as if the assignment is allowed to vary with the problem.

The binary classifiers are then evaluated on the testing data set. In Tables 2 and 3, we compare their performance against each other and the naive classifier, for contour directional features and for density features. These tables give the percentage of confusing pairs on which each classifier surpasses various min-recall thresholds. This evaluation is only carried out for the 528 confusing pairs for which both classes have at least ten samples of testing data.

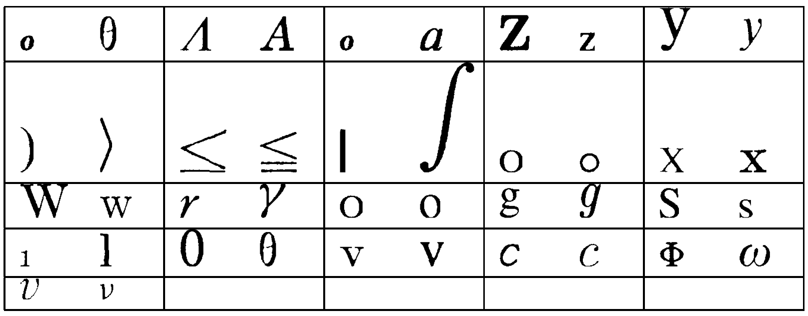


Fig. 3. Pairs on which the min-recall of linear SVM with directional features is at least 10% higher than that of the naive classifier.

Using contour directional features, each SVM kernel substantially exceeds the performance of the naive classifier. The Gaussian kernel falls short of the performance of the linear kernel. Feature choice matters greatly, perhaps because training samples are so scarce. With density features, the linear SVM performs slightly worse than with directional features, but more complicated kernels perform far worse, not even matching the naive classifier's benchmark. These kernels require more training time, so that fewer kernel parameter choices can be tested within the eight minute training limit. The linear SVM with directional features is efficiently chosen, trained, and utilized, and is as effective as the other classifiers, so we will use it as the basis for the analyses and multi-class experiments in the following sections.

The confusing pairs on which the linear SVM achieves the greatest improvement in min-recall over the naive classifier are shown in Figure 3. The most difficult pairs for SVM are shown in Table 4; as expected, many of these require large fractions of training vectors for support. Remarkably, many distinctions are adequately learned by the linear SVM without much training data. Figure 4 plots the min-recall for each confusing pair, against the number of training samples in the smaller class of the pair.

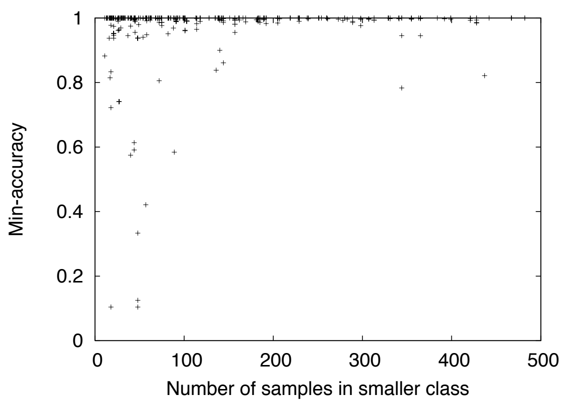


Fig. 4. Recall versus the number of training samples.

5 Multi-class classification

By starting with a fast classifier, we reduce our multi-class classification problem from 1,629 classes to the size of the confusion cluster of an Infty recognition result, which can vary as shown in Figure 2. Popular methods of combining binary SVM to perform multi-class classification are reviewed in [11], including a method based on one-versus-all classifiers, and two methods based on one-versus-one classifiers (the max-wins and directed acyclic graph approaches). Each approach has well-known drawbacks, and none is suited to utilize *a priori* information about the likelihood of alternatives, though the directed acyclic graph method requires an order for the candidates to be chosen, whose implications are far from obvious.

Any of these methods could be applied directly to a confusion cluster, but instead, we use a method that utilizes the ranking of alternatives in a confusion cluster, to make it likely that the most likely misrecognitions will be tested with an SVM.

For an Infty recognition result i, the confusion cluster $C(i)$ of misrecognition candidates is partially ordered by likelihood, as explained in Section 3. Let $C'(i)$ be the subset of alternatives $j \in C(i)$ for which a binary SVM comparing j and i was constructed. After the pure Infty engine recognizes a character as i, our method starts to apply the SVM for (j, i) for each $j \in C'(i)$, starting with the most likely j. When any j wins over i in the SVM classification, the testing is stopped, and j is reported as the classification. If no j wins, i is kept as the classification.

This method requires us only to train SVM on confusing pairs; other 1-versus-1 approaches would require us to train SVM on all pairs of letters that appear together in some confusion cluster. Of course, testing complexity is also linear in the number of letters in a cluster.

Without SVM, the pure Infty engine recognizes characters with 96.10% accuracy on our testing data set. Using SVM by this method, the recognition rate rises to 97.70%, so that the number of misrecognized characters falls by 41%.

When Infty makes the correct choice and our method does not, it always means that an SVM's decision was at fault. If neither Infty nor our method chooses correctly, three phenomena can explain the mistake. The SVM testing the Infty's choice against the right alternative may have chosen the wrong result when it was reached (we count the cases where an SVM was not trained, because of insufficient data, as such a case). The confusion of Infty's guess for the correct answer might not have occurred in the training data, so that the right alternative was not represented in the confusion cluster; we call this situation an "unprecedented mistake." The final alternative is called "shadowing." On an instance of testing data for which Infty guesses i, and the correct answer is k, we say that an SVM is "shadowed" if some other alternative j occurs before k in the confusion cluster, and j defeats i, so that the i versus k classifier is never run.

Altogether, the classification on the 70,637 testing samples may be synopsized as follows:

- Infty right, output right: 67,100
- Infty wrong, output right: 1,912
- Infty right, output wrong: 784
- Infty wrong, output wrong, SVM wrong or not trained: 399
- Infty wrong, output wrong, unprecedented mistake: 280
- Infty wrong, output wrong, SVM shadowed: 162

If shadowing happened frequently, our multi-class strategy would be inappropriate, but this data shows that it happens quite rarely.

6 Style distinction

One novel aspect of our single-character recognition problem is the distinction of a letter in Roman, italic, calligraphic, and blackboard bold styles, regardless of its font. The efficacy of SVM on this aspect of the problem is compared to that of other techniques in Table 5.

The decrease in the number of confusing pairs means that the SVM can distinguish certain styles with 100% accuracy that pose confusion to other classifiers. The total number of style mistakes decreases from Infty to SVM by a greater margin than the misrecognition rate overall.

With occasional mistakes, the naive classifier typically can distinguish calligraphic and blackboard bold from other styles. Its main weakness is the distinction of italic characters. The linear SVM shows significant improvement in this regard. In Figure 5, we display three italic pairs that are markedly improved with SVM.

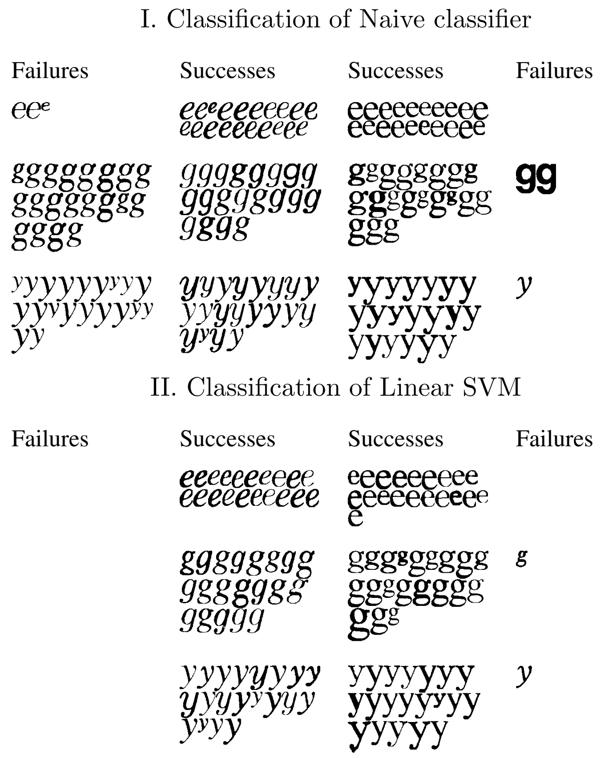

I. Classification of Naive classifier
Failures
Successes
Successes
Failures
II. Classification of Linear SVM
Failures
Successes
Successes
Failures

Fig. 5. Classification of the same letters in different styles.

The only case where linear SVM performed remarkably worse than the naive classifier was in the distinction of lower case italic l from script lower case ℓ.

7 Summary

We have demonstrated the effectiveness of SVM on a large multi-class problem, with many similar symbols and many classes with little training data. The SVM managed to learn many binary classifications well for which there was a paucity of training data. Though all SVM kernels provided about the same performance on directional features, the linear classifier had superior performance on density features. Generally, SVM trained on directional features performed marginally better than SVM trained on density features. The SVM excels at distinguishing styles of characters, particularly italic and non-italic variants, which are indistinguishable to simpler methods using the same sets of features.

We have integrated these SVM into the solution of a large multi-class problem, by testing only pairs of symbols mistaken by an existing OCR system. The complexity is low, and the most likely confused alternatives are preferred by the algorithm. The single-character misrecognition rate of the OCR system falls by 41% with the introduction of SVM. We note that we do not omit pairs often regarded as indistinguishable without size information (lower and upper case versions of C, O, P, S, U, V, X, and Z) in reporting our recognition rate.

Many of the mistakes that remain after the application of SVM represent characters that are truly indistinguishable without contextual information (such as the character's size relative to surrounding characters), or that represent degraded character images. We will try to improve the use of contextual information in Infty, and develop better methods for the treatment of touching and broken characters, in future work.

Acknowledgments

This work was supported by the Kyushu University 21st Century COE Program, "Development of Mathematics with High Functionality."

References

[1] Anderson, R. *Syntax-directed recognition of hand-printed two-dimensional mathematics.* PhD thesis, Harvard University, 1968.

[2] Chan, K.-F., and Yeung, D.-Y. Mathematical expression recognition: a survey. *IJDAR 3*, 1 (2000), 3–15.

[3] Chang, C.-C., and Lin, C.-J. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/%7Ecjlin/libsvm`.

[4] Chang, F., Lin, C.-C., and Chen, C.-J. Applying a hybrid method to handwritten character recognition. In *ICPR '04: Proceedings of the 17th International Conference on Pattern Recognition* (Washington, DC, USA, 2004), vol. 2, IEEE Computer Society, pp. 529–532.

[5] Cortes, C., and Vapnik, V. Support-vector networks. *Mach. Learn. 20*, 3 (1995), 273–297.

[6] DeBacker, S. Stable distributions supported on the nilpotent cone for the group G_2. In *The Unity of Mathematics; In Honor of the Ninetieth Birthday of I.M. Gelfand*, Progress in Mathematics, vol. 244. Birkhäuser, Boston, 2006.

[7] Dong, J.-X., Krzyzak, A., and Suen, C. An improved handwritten Chinese character recognition system using support vector machine. *Pattern Recogn. Lett. 26*, 12 (2005), 1849–1856.

[8] Eto, Y., and Suzuki, M. Mathematical formula recognition using virtual link network. In *ICDAR '01: Proceedings of the Sixth International Conference on Document Analysis and Recognition* (2001), IEEE Computer Society Press, pp. 430–437.

[9] Garain, U., Chaudhuri, B. B., and Ghosh, R. P. A multiple-classifier system for recognition of printed mathematical symbols. In *ICPR '04: Proceedings of the 17th International Conference on Pattern Recognition* (Washington, DC, USA, 2004), IEEE Computer Society, pp. 380–383.

[10] Hsu, C.-W., Chang, C.-C., and Lin, C.-J. A practical guide to support vector classification. `http://www.csie.ntu.edu.tw/%7Ecjlin/papers/guide/guide.pdf`, July 2003.

[11] Hsu, C.-W., and Lin, C.-J. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks 13* (2002), 415–425.

[12] Kimura, F., Wakabayashi, T., Tsuruoka, S., and Miyake, Y. Improvement of handwritten Japanese character recognition using weighted direction code histogram. *Pattern Recognition 30*, 8 (1997), 1329–1328.

[13] Malon, C., Uchida, S., and Suzuki, M. Separation of touching characters using DP matching. In *IEICE Technical Report PRMU2006: Proceedings of the IEICE Conference on Pattern Recognition and Machine Understanding* (2007), pp. 13–18.

[14] Meshesha, M., and Jawahar, C. Recognition of printed Amharic documents. In *ICDAR '05: Proceedings of the Eighth International Conference on Document Analysis and Recognition* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 784–788.

[15] Murakami, M., and Suzuki, M. Improvement of mathematical structural analysis by Center-Band. In *IEICE Technical Report PRMU2001-270 (2002-03)*, pp. 203–210.

[16] Suzuki, M. InftyCDB-3: a ground truthed database of words/formulae images, third distribution. `http://www.inftyproject.org/en/database.html`.

[17] Suzuki, M., Kanahori, T., Ohtake, N., and Yamaguchi, K. An integrated OCR software for mathematical documents and its output with accessibility. In *Computers helping people with special needs, 9th International Conference ICCHP 2004, Paris* (July 2004), Lecture Notes in Computer Science 3119, Springer, pp. 648–655.

[18] Suzuki, M., Malon, C., and Uchida, S. Databases of mathematical documents. *Research Reports on Information Science and Electrical Engineering of Kyushu University 12*, 1 (2007), 7–14.

[19] Suzuki, M., Tamari, F., Fukuda, R., Uchida, S., and Kanahori, T. Infty: an integrated OCR system for mathematical documents. In *DocEng '03: Proceedings of the 2003 ACM symposium on Document engineering* (New York, NY, USA, 2003), ACM Press, pp. 95–104.

[20] Suzuki, M., Uchida, S., and Nomura, A. A ground-truthed mathematical character and symbol image database. In *ICDAR '05: Proceedings of the Eighth International Conference on Document Analysis and Recognition* (Washington, DC, USA, 2005), IEEE Computer Society, pp. 675–679.

[21] Tapia, E., and Rojas, R. Recognition of on-line handwritten mathematical formulas in the E-Chalk system. In *ICDAR '03: Proceedings of the Seventh International Conference on Document Analysis and Recognition* (Washington, DC, USA, 2003), IEEE Computer Society, pp. 980–984.

[22] Toshihiro, K., and Masakazu, S. Detection of matrices and segmentation of matrix elements in scanned images of scientific documents. In *ICDAR '03: Proceedings of the Seventh International Conference on Document Analysis and Recognition* (Washington, DC, USA, 2003), IEEE Computer Society, p. 433.

[23] Uchida, S., Nomura, A., and Suzuki, M. Quantitative analysis of mathematical documents. *International Journal on Document Analysis and Recognition 7*, 4 (2005), 211–218.

[24] Vapnik, V. *The nature of statistical learning theory.* Springer-Verlag New York, Inc., New York, NY, USA, 1995.

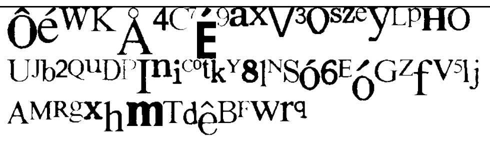
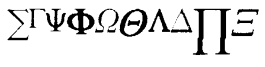
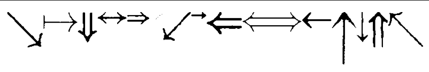
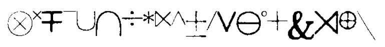
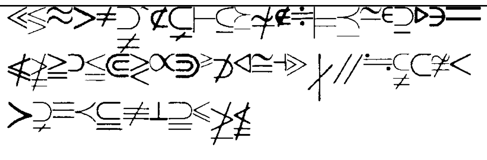
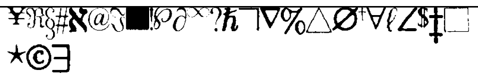
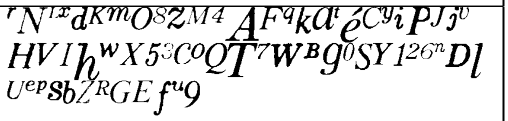
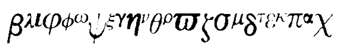
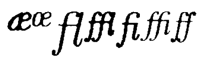
Upright Latin	
Upright Greek	
Calligraphic	
Blackboard Bold	
Punctuation	
Brackets	
Accents	
Arrows	
Binary Operators	
Relational Operators	
Big Symbols	
Other Symbols	
Italic Latin	
Italic Greek	
Italic Ligature	
Upright Ligature	

Table 1

Classes represented in InftyCDB-3-A.

Min-recall	Naive	SVM Linear	SVM Gaussian	SVM Cubic
> 0	100.00%	100.00%	100.00%	100.00%
$> .5$	98.67%	99.05%	99.05%	99.05%
$> .6$	98.30%	98.48%	98.67%	98.48%
$> .7$	97.35%	98.30%	98.30%	98.30%
$> .8$	95.27%	97.35%	97.54%	97.35%
$> .9$	93.75%	95.83%	95.64%	95.64%
$> .95$	90.91%	92.99%	93.18%	92.61%
$> .97$	84.28%	90.34%	89.77%	89.96%
$> .99$	73.30%	84.28%	82.95%	84.28%
$> .995$	66.86%	78.22%	74.62%	77.84%
$> .999$	56.82%	69.13%	64.39%	69.89%

Table 2

SVM performance on confusing pairs, using contour directional features.

Min-recall	SVM Linear	SVM Gaussian	SVM Cubic
> 0	100.00%	99.60%	96.21%
$> .5$	98.86%	97.18%	94.51%
$> .6$	98.48%	96.98%	94.32%
$> .7$	98.11%	96.18%	92.99%
$> .8$	97.16%	94.97%	91.10%
$> .9$	94.70%	91.75%	88.07%
$> .95$	91.10%	86.92%	83.33%
$> .97$	88.83%	80.08%	78.22%
$> .99$	82.58%	71.03%	70.83%
$> .995$	78.03%	67.00%	66.29%
$> .999$	67.80%	55.33%	51.14%

Table 3

SVM performance on confusing pairs, using density features.

Min-recall	x-recall	y-recall	x	y	Max SV fraction
0.1042	0.1042	0.9553	***o***	*O*	.3548
0.1042	0.1042	1.0000	***o***	*O*	.1935
0.1250	1.0000	0.1250	0	***o***	.1935
0.3333	0.3333	1.0000	***o***	*a*	.3548
0.4211	0.4211	0.9306	**Z**	z	.2934
0.5750	0.9907	0.5750	*l*	!	.0698
0.5843	0.5843	0.8704	v	**V**	.1718
0.5909	0.5909	1.0000	ℓ	*l*	.1400
0.6136	0.6136	1.0000	ℓ	*p*	.1200
0.7222	0.7222	0.9297	*O*	***o***	.3768
0.7407	1.0000	0.7407	O	o	.1236
0.7407	0.9826	0.7407	O	o	.0356
0.7407	0.9943	0.7407	0	o	.0300
0.7833	0.9535	0.7833	O	O	.2556

Table 4

Pairs with linear SVM min-recall below .80.

	Naive	Infty	SVM
Total number of confused pairs	315	321	256
Confused pairs representing style mistakes	46	51	37
Total number of misrecognitions	3,832	2,753	1,625
Style recognition errors	254	219	116

Table 5

Style misrecognitions on testing data