Optical Character Recognition of Printed Mathematical Symbols using A Hierarchical Classifier

Jason Ranger, Fei Wang, Howard Cheng[1]
Department of Mathematics and Computer Science
University of Lethbridge, 4401 University Drive, Lethbridge, Alberta, Canada
jason.ranger@uleth.ca, f.wang@uleth.ca, howard.cheng@uleth.ca
[1]corresponding author

Abstract— *In this paper, we examine the effectiveness of a hierarchical approach for the optical character recognition of printed mathematical symbols in a document. We make use of a number of features extracted from the character bitmaps, and experiments were performed to determine the effectiveness of these features in identifying the actual character. The characters are first classified using k-means clustering with a subset of these features into smaller groups, and the final recognition is performed using support vector machines (SVM) in each group. The recognition rate of this approach is over 97% and compares favourably with existing methods.*

Keywords: Optical character recognition, k-means clustering, support vector machines.

1. Introduction

The problem of optical character recognition of "standard" text documents has been well studied. The study of recognizing printed mathematical symbols, however, is relatively recent [2], [9], [10]. This problem has gained importance in recent years because of interests in converting printed mathematical documents into electronic form. For example, one of the more successful projects is the Infty project [9], [10].

One key step in the conversion of printed mathematical documents into electronic form is the optical character recognition (OCR) of individual mathematical symbols. A number of challenges in mathematical OCR include [2], [6]:

- some symbols may play different roles in different contexts. For example, a dot may be multiplication, decimal point, or other;
- some groups of distinct symbols are very similar to each other in appearance. For example, Latin v and Greek ν;
- different styles of the same symbol (e.g. italic, bold, calligraphic, etc.) need to be distinguished as different styles often carry different meanings in the document;
- it is more difficult to correct misrecognition as there is often no easy "spelling" or "grammar" check that can be performed.

In this work, we examine a hierarchical approach to OCR of printed mathematical symbols. We restrict our work only to the OCR step, so that no attempt is made to analyze the recognition results of close-by symbols in order to improve the accuracy. Contextual information such as the sizes, locations, and darkness of symbols in relation to other symbols is not used. Through experiments, we select a subset of features to be used by a k-means clustering to group similar symbols into cluster. For clusters that contain more than one type of symbols, a separate set of features is used to train support vector machines to identify the symbol. We show that the recognition rate of our approach is over 97%. This is similar to the recognition rates of other existing approaches, but our approach is more adaptive and can be more efficient.

2. Previous Works

In [6], [7], support vector machines (SVM) were used to improve classification by an existing OCR system used by the InftyReader engine [6] or a naïve classifier [7]. Classification results were used to produce a "confusion matrix" which represents symbols that are misclassified. "Confusion clusters" are then formed consisting of symbols that are incorrectly recognized as the same symbol. For each cluster, a set of SVMs is trained to identify the symbols in the cluster. Although most of these clusters have fewer than 5 distinct types of symbols, some have as many as 26. The large number of distinct types lead to the need to apply non-standard multi-class SVM methods [6].

Features need to be extracted from each symbol for the purpose of training as well as classification. The bitmap of each character is first divided into a 3×5, 5×5, or 5×3 mesh depending on its aspect ratio. In each grid cell of the mesh, the chain code of the boundary of the character is followed and a histogram is built on the frequencies of vertical, horizontal, diagonal-1, and diagonal-2 codes. This results in 221-dimensional feature vectors[1]. Insignificant blocks' coordinates are changed to zero. The method has a recognition rate of 97.70%.

There are other approaches that attempt to combine multiple classifiers to perform recognition. See, for example, [4]. It differs from the current work in that the results from multiple

[1]The classifier in [7] used meshes of different sizes, resulting in 160-dimensional feature vectors.

classifiers are combined using a variety of methods. The recognition rate is around 93%. The approach used by [1] has a recognition rate of around 95%.

3. Feature Selection

In this study, we chose a number of well-known features to examine their suitability for the OCR of printed mathematical symbols. Before extracting the features, each character is segmented into a bounding box and normalized to have the same size. It is assumed that this step has already been done and is not examined in this paper.

Below, we list the features that were finally selected. For a more detailed description of these features, see, for example, [3].

Area:	number of foreground pixels.
Aspect Ratio:	quantized into one of three categories: "short", "square", or "tall".
Centroid:	centroid of the shape.
Centroid Distance:	the maximum, minimum, and mean distance of boundary pixels to the centroid.
Crossings:	number of times each scanline crosses a boundary.
Density:	the bitmap is divided into 16 zones and the number of foreground pixels in each zone is the density of the zone.
Diameter:	maximum distance between pixels in the foreground.
Directional Features:	mesh features used in [7]. There are 160 dimensions.
Distance Transform:	the average distance of a pixel from the foreground to the background.
Feature Points:	after the shape has been skeletonized, the number of endpoints, T-intersections, and other intersections in the skeleton.
Holes:	number of holes in the symbol.
Perimeter:	number of pixels on the boundary.
Profile:	the distance from a background pixel on the boundary of the bitmap to a foreground pixel. There are 64 background pixels used.
Projection:	number of pixels on each scanline. There are 32 scanlines.
Shape Complexity:	ratio of the average distance transform and area.
Symmetry:	measure of how symmetric the shape is in vertical and horizontal directions.
Shape Number:	normalized chain code of the boundary.
Hu Moments.	Normalized to be translation, scale, and rotation invariant.

4. Test Data

In our experiments, we use the InftyCDB-3 [8] database to train and test our classifier. This database is used in our testing instead of the InftyCDB-1 database [11] because we are only interested in the characters and not the structural information of the characters in the expressions. The database consists of bitmap images each containing an image of a single symbol. The symbols come from a variety of publications. Ground truth for each symbol is provided, so that a desired label is known.

However, the ground-truth labels have some interesting properties. In some cases, distinct types of symbols may be labelled the same. For example, a horizontal line may be minus, overbar, underbar, and so on. When the character is taken out of context, it is impossible to distinguish them.

Some symbols that look different but are not distinguished in mathematical usage are also labelled the same way (e.g. ϵ and ε, slanted and italic). This means that some symbols may actually be represented by multiple distinct clusters in the feature space.

Finally, some symbols that look almost the same out of context are labelled differently in this database. For example, some uppercase and lowercase letters (e.g. 'C' and 'c', 'S' and 's') are very difficult (even by human) to distinguish out of context.

To ensure that there are enough data to train and evaluate our classifiers, characters with too few samples are not used in the training and testing of our classifier. We randomly select a portion of the samples of each remaining symbol for training. The rest of the samples are used for testing.

5. Hierarchical Classifier

The proposed approach consists of classifiers at a number of different levels. For training, a set of features is used to put each symbol into an appropriate bin. For each bin, k-means clustering is applied to a second set of features to obtain clusters of symbols. Finally, a third set of features is used to train support vector machines (SVMs) in each clusters to identify individual symbols. For classifying symbols, the appropriate features are extracted. The appropriate bin is computed, and the closest cluster in that bin is found. The SVMs associated with that cluster is used for the final recognition. The structure of the classifier is shown in Figure 1.

The main advantage of the hierarchical approach is when there are symbols that are represented by a number of distinct clusters in the feature space. This situation occurs often as the same symbol may be corrupted in a few different ways, leading to distinct clusters (e.g. some of the digit "8" may be broken, resulting in fewer than 2 holes). It can also happen because a symbol may appear differently depending on the font used, and we want to recognize these symbols as the same (e.g. ϵ and ε). If we consider all of these symbols as one

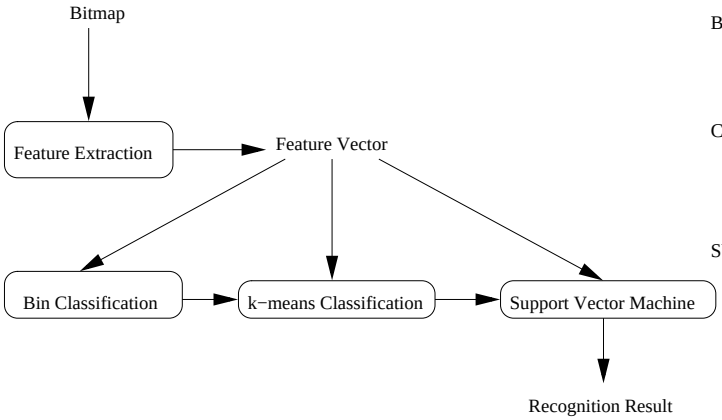

Fig. 1: Structure of hierarchical classifier.

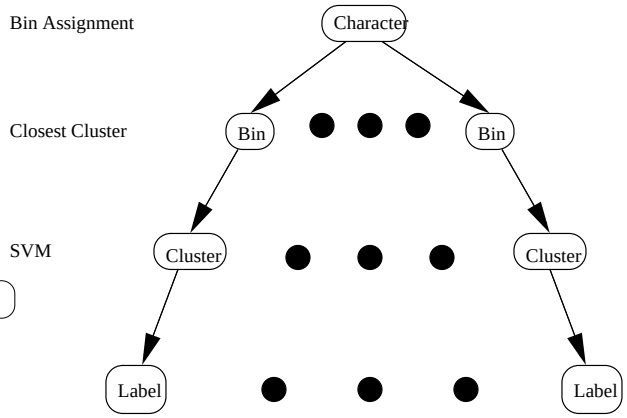

Fig. 2: Tree representation of hierarchical classifier.

type in a classification scheme (e.g. k-means and SVM), the clusters for each type will be large and may even overlap with each other. On the other hand, if the top-level classifier can partition these clusters into tighter subclusters, lower-level classifiers will be more accurate. The hierarchical approach is also beneficial for using SVMs since SVMs are inherently two-class methods. Having fewer distinct types of symbols in each cluster will make it easier to train SVMs for the cluster, so that sophisticated multi-class SVM schemes are not needed. That is the reason for using SVMs only at the bottom level.

We performed experiments in choosing the subset of features at each level. An exhaustive search on the possible choices of features and classifiers at each level is performed. Based on this search, we describe the optimal arrangement of the classifiers below. It is interesting to note that we did not find the directional features used in [6], [7] helpful for classification compared to the other features we examined.

At the top level, the *Holes* and *Aspect Ratio* features are used to classify each symbol into an appropriate bin. Through experiments, it was discovered that an optimal way to classify symbols into bins is based on *Holes* and *Aspect Ratio*. The feature *Holes* is classified into three groups—0 holes, 1 hole, and 2 or more holes. For *Aspect Ratio* it was determined that it was best to classify the symbols into "short" and "tall", with "square" being grouped with "tall" as well. This results in 6 bins of symbols after the first step.

For each of the 6 bins, k-means clustering is performed using the *Distance Transform*, *Density*, and *Profile* features. For this clustering k is set to be the number of different classes of symbols in each bin. The centroid of each resulting cluster is computed. During classification, the closest cluster (with respect to the centroid) is chosen. There are in fact few clusters with 2 or more types of symbols. 80.0% of the clusters have only 1 type of symbol, 14.8% of the clusters have two types, 4.3% of the clusters have 3 types, and the remaining clusters have 4 types. No clusters have more than 4 types of symbols.

For each of the clusters that has more than one type of symbols, we build SVM classifiers on small subsets of the remaining features to identify the individual type of symbols using one-vs-one SVMs and linear kernels [5]. In our experiments, only two or three additional features are needed for the SVMs to distinguish the different symbols, although the subset of features used may be different for each cluster.

The resulting hierarchical classifier can be represented as a tree of height 3, with the nodes at the three levels representing the bins, the clusters, and finally the individual symbols (Figure 2). A classifier is associated to each non-leaf node that uses a subset of features to classify the incoming character into a leaf node. Since the subset of features needed to recognize a character may be different due to the adaptive nature of our classifier, it is computationally more efficient to extract features incrementally as needed. Only "difficult" symbols require more work. For many symbols, there is no need to apply SVM classifier (and hence to compute additional features) because the corresponding cluster uniquely identifies the symbol.

There are some similarities between our approach and that of [6], [7]. Their approach first uses a classifier to produce a potential label, and the result is corrected by SVMs. This can be considered a hierarchical method in which the confusion clusters produced by the first classifier are analogous to the clusters we obtain with k-means clustering. An important difference in our approach is that the subset of features used by the SVMs in each cluster is adaptive to each cluster, while their approach uses the same set of features for each cluster. Since each cluster may have different statistical properties in the feature space, our approach is adaptive in that it chooses the most effective subset of features to distinguish the symbols in each cluster. In addition, some of the features chosen for the clusters in our approach are significantly simpler than the high-dimensional mesh features used in [6], [7], so our approach can be more efficient in both the training and classifying steps. Furthermore, our clusters generally

(a) (b)

Fig. 3: A typical pair of characters misclassified. (a) Uppercase 'X'. (b) Lowercase 'x'.

have significantly fewer distinct types of symbols compared to the confusion clusters in [6], [7], facilitating the use of SVMs in these clusters.

6. Experimental Results

We performed exhaustive search to determine the optimal arrangement of classifier described above. We varied the number of levels of classifiers and the different subsets of features used at each node. Each choice of parameters was evaluated by first training the classifiers with the training portion of the database and then classifying the testing portion. The choice leading to the highest recognition rate was chosen.

Our recognition rate on the testing portion of the InftyCDB-3 database was 95.6%, which appears lower than some other existing methods at first glance. Of the 4.4% of symbols incorrectly recognized, 1.6% of these symbols are classified incorrectly into the wrong bin at the first step. In these cases, no training samples with the correct label were available in the bin. This shows a disadvantage of the hierarchical approach—an incorrect classification at the top level (possibly because of lack of training samples) are impossible to correct at lower levels.

The results of our classifier are actually better on closer examination of the misclassified symbols. Many of the misclassified symbols are those symbols in the InftyCDB-3 that are visually very similar out of context but have different labels. For example, uppercase and lowercase 'c', 'o', 's', 'v', 'x', 'w' are commonly mislabelled (Figure 3). Without surrounding context, it is impossible to distinguish them even for human viewers. When these results are discarded (they are also discarded in [7], for example), the recognition rate of our classifier is 97.8%, which is amongst the best in existing work [1], [4], [6], [7].

Finally, we also see from the classification results that labels consisting of different distinct symbols are indeed classified correctly. This justifies the use of the hierarchical approach.

7. Conclusions

In this paper, we examined a wide range of features and performed an exhaustive search to determine an optimal arrangement in a hierarchical classifier for printed mathematical symbols. Our final classifier performs favourably against other existing classifiers. We believe that the results can be further improved by preprocessing as well as context information, and we are currently investigating this possibility.

References

[1] F. Álvaro and J.A. Sánchez. Comparing several techniques for offline recognition of printed mathematical symbols. In *20th International Conference on Pattern Recognition (ICPR)*, pages 1953–1956, aug. 2010.

[2] K.-F. Chan and D.-Y. Yeung. Mathematical expression recognition: a survey. *International Journal on Document Analysis and Recognition*, 3:3–15, 2000. 10.1007/PL00013549.

[3] L. Costa and R. Cesar, Jr. *Shape Analysis and Classification: Theory and Practice*. CRC Press, Inc., Boca Raton, FL, USA, 1st edition, 2000.

[4] U. Garain, B.B. Chaudhuri, and R.P. Ghosh. A multiple-classifier system for recognition of printed mathematical symbols. In *Proceedings of the 17th Intl. Conf. on Pattern Recognition (ICPR'04)*, pages 380–383, 2004.

[5] C.-W. Hsu and C.-J. Lin. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, mar 2002.

[6] C. Malon, S. Uchida, and S. Masakazu. Mathematical symbol recognition with support vector machines. *Pattern Recognition Letters*, 29(9):1326–1332, 2008.

[7] C. Malon, S. Uchida, and M. Suzuki. Support vector machines for mathematical symbol recognition. In *Structural, Syntactic, and Statistical Pattern Recognition*, volume 4109 of *Lecture Notes in Computer Science*, pages 136–144. Springer Berlin Heidelberg, 2006.

[8] The Infty Project. `http://www.inftyproject.org/`.

[9] M. Suzuki, T. Kanahori, N. Ohtake, and K. Yamaguchi. An integrated OCR software for mathematical documents and its output with accessibility. In *Computers Helping People with Special Needs*, volume 3118 of *Lecture Notes in Computer Science*, pages 648–655. Springer Berlin Heidelberg, 2004.

[10] M. Suzuki, F. Tamari, R. Fukuda, S. Uchida, and T. Kanahori. INFTY: an integrated OCR system for mathematical documents. In *Proceedings of the 2003 ACM Symposium on Document Engineering*, DocEng '03, pages 95–104, 2003.

[11] M. Suzuki, S. Uchida, and A. Nomura. A ground-truthed mathematical character and symbol image database. In *Proceedings of the Eighth International Conference on Document Analysis and Recognition (ICDAR '05)*, pages 675–679, 2005.