Identification of embedded mathematical formulas in PDF documents using SVM

Xiaoyan Lin\textsuperscript{*}, Liangcai Gao\textsuperscript{†}, Zhi Tang\textsuperscript{a,b}, Xuan Hu\textsuperscript{c} and Xiaofan Lin\textsuperscript{d}
\textsuperscript{a} Institute of Computer Science and Technology, Peking University, Beijing, China; 
\textsuperscript{b} State Key Laboratory of Digital Publishing Technology, Beijing, China; 
\textsuperscript{c} College of Software, Beihang University, Beijing, China; 
\textsuperscript{d} Vobile Inc, Santa Clara, California, USA

ABSTRACT

With the tremendous popularity of PDF format, recognizing mathematical formulas in PDF documents becomes a new and important problem in document analysis field. In this paper, we present a method of embedded mathematical formula identification in PDF documents, based on Support Vector Machine (SVM). The method first segments text lines into words, and then classifies each word into two classes, namely formula or ordinary text. Various features of embedded formulas, including geometric layout, character and context content, are utilized to build a robust and adaptable SVM classifier. Embedded formulas are then extracted through merging the words labeled as formulas. Experimental results show good performance of the proposed method. Furthermore, the method has been successfully incorporated into a commercial software package for large-scale e-Book production.

Keywords: Mathematical formula recognition, formula identification, embedded formulas, PDF documents

1. INTRODUCTION

Nowadays an increasing number of scientific documents are available in PDF format, which can greatly facilitate document exchange and printing. As a common component of the scientific documents, mathematical formulas are still recognized at accuracy too low to be useful in many practical applications. Identifying the regions of mathematical formulas is the first step in this task. Mathematical formulas are classified into two categories: 1) isolated formulas printed in separate lines; 2) embedded formulas mixed with ordinary text. Existing isolated formulas identification approaches are rule-based\textsuperscript{1-3} or machine learning-based\textsuperscript{4}. Since the isolated formulas exhibit distinct geometric layout features, it is relative easier to identify them. It is observed that some isolated formula identification techniques provided rather high accuracy\textsuperscript{1-3}. In contrast, the embedded formula identification is more challenging, because the embedded formulas are generally short expressions, which are difficult to discriminate from ordinary text. Consequently, the lower accuracy is expected for embedded formula identification compared with isolated formula identification.

As far as we know, current embedded formula identification methods are all rule-based and only focus on image-based documents. According to the types of features used, the existing methods can be classified into two categories: character-based methods and layout-based methods:

Character-based methods\textsuperscript{5-9} locate embedded expressions mainly through identifying special math symbols (e.g., “=”, “+”, “<”, etc.) and applying specific context propagation rules from these symbols. Lee et al.\textsuperscript{3} utilize recognized math symbols as seeds to generate geometric trees of formulas and heuristically attach adjacent symbols including those subscribing or matrix structures. Kacem et al.\textsuperscript{9} use fuzzy logic to identify some mathematical symbols and then propagates the context around the math symbols to detect the embedded formula areas. Inoue et al.\textsuperscript{6} extract embedded formulas in Japanese scientific document based on the assumption that math symbols are either rejected or recognized with low confidence by OCR. Although character-based methods are straightforward, it requires significant implementation effort since the rules are set according to each type of formulas and different cases. Moreover, the rules

\textsuperscript{*} linxiaoyan@pku.edu.cn
\textsuperscript{†} glc@pku.edu.cn (Liangcai Gao is the corresponding author)
are often too rigid to adjust to different applications and new cases. For embedded formulas with unknown math symbols or in unknown formats or types, character-based methods tend to be ineffective. In addition, given the wide variety of formula formats, it is almost impossible to find all of rules for embedded formula types.

Layout-based methods\textsuperscript{1-4} extract embedded formulas mainly by utilizing the layout features. To our best knowledge, the published layout-based methods are all rule-based. Garain et al.\textsuperscript{1,2} identify the embedded formulas by constructing rules based on some common typographical conventions followed in typing mathematical expressions. Several crucial thresholds are set based on statistics, which can be very sensitive to the ratio of embedded formulas and ordinary text. Jin et al.\textsuperscript{3} and Chowdhury et al.\textsuperscript{4} extract embedded formulas by detecting two-dimensional structures based on the symbols’ projection characteristics. This method can only detect two-dimensional embedded formulas. The embedded formulas are generally short expressions, thus some elements of the embedded formulas (e.g., variables and functions) share similar layout features with ordinary text. Therefore, it can be ineffective to discriminate embedded formulas from ordinary text by constructing the rule-based quantitative models of layout features. First, in rule-based methods it is hard to set appropriate thresholds and parameters because they are sensitive to document types and formats. Moreover, rule-based methods rely heavily on heuristics which are usually not adaptive enough to deal with the varieties of embedded formula types. Thus, most of the layout-based methods do not yield satisfactory results.

To solve the above problems, we present a SVM-based method to achieve robust and adaptable embedded formula identification. On the one hand, we exploit machine learning techniques to identify embedded formulas while overcoming the problems (e.g., threshold setting) associated with traditional layout-based methods. On the other hand, various features (layout, character and context) are explored. Rather than constructing rules according to the features for each type of formula, the features are quantized and utilized as the feature vector to build SVM classifier. The trained classifier then enables embedded formulas identification in a more flexible and adaptable fashion.

The rest of the paper is organized as follows. Section 2 gives an overview of the whole system. Section 3 introduces the SVM-based embedded formula identification method. The experimental results are shown in Section 4, and we draw conclusions and discuss future work in Section 5.

2. OVERVIEW

In our earlier work\textsuperscript{13}, a system for mathematical formula identification in PDF documents is presented, as shown in Figure 1. The workflow of our method is as follows: First, preprocessing is employed to match the different types of content objects (text, image, and graph) parsed from PDF documents to the mathematical elements. And information (e.g., bounding box, baseline and font, etc) of the mathematical elements is obtained. Next, text lines are extracted. Then, isolated formula lines are identified by a hybrid method. Finally, a rule-based method is adopted to extract the embedded formulas. Experimental results show high precision for isolated formula identification and lower precision for embedded formula identification. In this paper, we focus on improving the embedded formula identification (the highlighted module in Figure 1) and propose techniques to detect the embedded formulas based on SVM classification rather than rules. In this work we assume that the input of the proposed method is a set of text lines that are not recognized as isolated formulas or ordinary text lines in the upstream steps of the system.

![Workflow of the formula identification](Figure 1)

Embedded mathematical formulas covered by our approach include not only two-dimension mathematical expressions (e.g., superscript, subscript, fraction, etc.), but also one-dimension mathematical expressions (e.g., variables,
functions, explicit expressions, etc.). In this paper the PDF documents are assumed to be in English rather than other languages (e.g., Chinese or Japanese), since mathematical formulas are mostly composed of Latin characters and math symbols, which are relative easy to distinguish from Chinese or Japanese characters by checking the character’s Unicode.

As shown in Figure 2, our embedded mathematical formula identification method consists of four steps: 1) Word segmentation; 2) Feature extraction; 3) SVM-based classification; 4) Formula region merging. Step 1 is to segment the text lines into words. Feature vector of each word is generated in Step 2. In Step 3, a SVM-based classifier is trained and utilized to predict if a word is a fragment of the embedded formula or an ordinary text word. The embedded formulas are then extracted by merging the regions of the successive words labeled as formula fragments in Step 4.

![Workflow of the SVM-based embedded formula identification](image)

Figure 2. Workflow of the SVM-based embedded formula identification

3. METHODOLOGY

3.1. Word segmentation

This step is to segment text lines into separated words. Since we focus on PDF documents that contain perfect typesetting information, the gaps between adjacent words can be very helpful. We employ a simplified word segmentation algorithm based on the word gaps and special separators. We traverse all the symbols in each line from left to right, and if one of the following two requirements is satisfied, a word is selected:

1. If the gap between successive characters is larger than a threshold (called word gap), a word should be selected. This threshold is obtained adaptively: First, a gap histogram is formed according to a set of predefined discrete gap ranges. Then, the second highest peak of the histogram is calculated and the corresponding discrete gap is chosen as the word gap. It is worth noting that the discrete gap corresponding to the highest peak of the histogram is actually the character gap within words.

2. If the current character is one of the special separators, a word should be selected. And the character itself should be also selected as a new word. We defined two categories of symbols as the special separators. One category is the punctuations (e.g., comma, period, dash, colon, etc.), and the other is the parentheses (e.g., ‘(’, ‘)’, ‘[’, ‘]’, etc.). Because each character’s Unicode has been parsed from PDF document, we can simply detect these separators through looking up the predefined special separator dictionary.

Through the above mentioned steps, both ordinary words will be selected and the embedded formulas will be segmented into irregular fragments. An example of word segmentation result is shown in Figure 3. It can be observed that the geometric layout features of the ordinary text words seem to be very different from those of the formula fragments. For instance, the font sizes of the constituent symbols in an ordinary text word always remain the same, whereas the font size of each symbol in a formula fragment varies a lot. However, at the symbol level or line level, the layout features of ordinary text fragments are very similar to those of formula fragments. Based on this observation, we define word-specific features to identify the embedded formulas in the next section.
3.2. Feature extraction

For each word, 12 features are extracted. As shown in Table 1, these features can be classified into three categories: geometric layout features, character features and context features. The first seven features are the layout features and the next three features are character features. The rest two features are the context features defining the context of the preceding and the following neighbors of the current word. To achieve better classification performance, each element’s value is normalized to \([-1, 1]\).

Table 1. Features of a word

<table>
<thead>
<tr>
<th>Features</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Geometric layout features</strong></td>
<td></td>
</tr>
<tr>
<td>V-Fontsize</td>
<td>Variance of the font size of the symbols in a word.</td>
</tr>
<tr>
<td>V-Position</td>
<td>Variance of the Y-coordinates of the baseline of the symbols in a word.</td>
</tr>
<tr>
<td>V-Space</td>
<td>Variance of the space of the bounding box of the symbols in a word.</td>
</tr>
<tr>
<td>V-Width</td>
<td>Variance of the width of the bounding box of the symbols in a word.</td>
</tr>
<tr>
<td>V-Height</td>
<td>Variance of the height of the bounding box of the symbols in a word.</td>
</tr>
<tr>
<td><strong>D-Purity</strong></td>
<td>Degree of the symbols in a word that belong to the same type. (Definition in detail is in Section 3.2.1.)</td>
</tr>
<tr>
<td>P-English</td>
<td>Percentage of the English characters of a word.</td>
</tr>
<tr>
<td><strong>Character features</strong></td>
<td></td>
</tr>
<tr>
<td>I-Math</td>
<td>Whether at least one of the specific mathematical entities appears in the word. Specific mathematical entities include relation operators, arithmetic operators, Greek letters, and math functions, etc.</td>
</tr>
<tr>
<td>T-Leftmost</td>
<td>Type of the leftmost symbol of a word. (Four symbol types are defined in Section 3.2.2.)</td>
</tr>
<tr>
<td>T-Rightmost</td>
<td>Type of the rightmost symbol of a word. The type of the rightmost symbol is defined as the same as the leftmost symbol’s.</td>
</tr>
<tr>
<td><strong>Context features</strong></td>
<td></td>
</tr>
<tr>
<td>T-LeftAdjacent</td>
<td>Type of the rightmost symbol of the previous word of the current word. The definition of symbol type is the same as the definition in T-Leftmost.</td>
</tr>
<tr>
<td>T-RightAdjacent</td>
<td>Type of the leftmost symbol of the following word of the current word. The definition of symbol type is the same as the definition in T-Leftmost.</td>
</tr>
</tbody>
</table>
3.2.1. Geometric layout features

The geometric layout features of a word are defined based on the following observations:

1. The font size or baseline (in Y-coordinates) of the symbols of a formula fragment varies a lot, such as, “a” and “x”, etc. However, the variances of font size and baseline of the symbols of an ordinary text word are almost equal to zero. To measure the variances of font size and baseline of the symbols in a word, \( \text{V-Fontsize} \) and \( \text{V-Position} \) are defined.

2. The spaces between successive symbols of an ordinary text word are regular, whereas the spaces between successive symbols of a formula fragment change a lot, e.g., “\( \sqrt{a+b} \)”. \( \text{V-Space} \) is defined to measure the variance of spaces between successive symbols of a word.

3. The widths or heights of the symbols of an ordinary text word stay relative stable, but the symbols of a formula fragment have different widths and heights in most cases, e.g., “\( \sum a_i \)” and “\( \left[ \frac{1}{x} \right] \)”, etc. Therefore, \( \text{V-Width} \) and \( \text{V-Height} \) are defined to describe the variances of the widths and heights for the symbols of a word.

4. The symbols in the ordinary text words are all English characters, while the formula fragments are composed of both not only English characters but also math symbols. To evaluate the degree of the symbols in a word belonging to the same type, \( \text{D-Purity} \) is defined. Symbols are classified into categories, namely English and non-English. Let \( P_E \) be the percentage of the English characters of a word and \( P_{\text{ne}} \) be the percentage of the non-English characters of a word, and \( \text{D-Purity} \) is defined as:

\[
\text{D-Purity(Word)} = -P_E \log_2(P_E) - P_{\text{ne}} \log_2(P_{\text{ne}})
\]

5. A straightforward idea is that the more English characters a word contains, the more it is likely to be an ordinary text word. This feature may be effective to discriminate ordinary text words from formula fragments. Thus, \( P_{\text{English}} \) is computed as one of the layout features of a word.

3.2.2. Character features

The previous embedded formula identification methods\(^5\)\(^-\)\(^10\) all apply character features to construct rules for each type of formulas. In order to avoid the drawbacks associated rule-based methods, we exploit the character features as a part of the feature vector to train classifiers automatically. Three character features are defined based on the following observations:

1. If there exists any specific mathematical entity in a word, namely math functions (e.g., sin, cos, etc) or math symbols (e.g., operators, Greek characters, etc.), the word must be a formula fragment. Hence, \( J-Math \) is defined to indicate whether any specific mathematical entities appear in the word. The specific mathematical entities are defined in a math entity dictionary.

2. The leftmost and rightmost symbols of the word may influence detecting the optimal boundaries of the embedded formulas. In particular, when the leftmost or rightmost symbol is one of the mathematical operators (e.g., “\( \int \)”, “\( + \)”, “\( = \)”, etc.), the operand domains of these operators offer reliable information to detect the boundaries. Operand domains indicate whether the context content of the current word is impossible or likely to be math fragments. For instance, if the rightmost symbol of the current word is “\( = \)”, the right adjacent word is very likely to be a formula fragment. According to the types of operand domains, the symbols are categories into four types, namely, non-math symbols, math symbols with no operand domain (e.g., Greek characters), unary math arithmetic/relation operators (e.g., “\( \int \)”, “\( \Sigma \)”) and binary math arithmetic/relation operators. \( T-\text{Leftmost} \) and \( T-\text{Rightmost} \) are defined to represent the types of the leftmost and rightmost symbols of the word.

3.2.3. Context features

Context depicts the relation and influence between the adjacent words (or characters). The context features are very important for detecting the boundaries of the embedded formulas. In rule-based methods, context features are exploited to construct propagation rules according to each type of mathematical symbols. Since there are many types of mathematical formulas and the formats of them vary a lot, the rules are generally inadequate to cope with the wide range of cases. To make use of the sequential information of the words to improve the classification of each word, context features are defined as two elements of the feature vector. We consider the types of adjacent symbols before and after the
current word as the context feature. It is based on the observation that the adjacent symbols before and after the current word provide meaningful information to decide whether the current word is a formula fragment or not. For instance, if the adjacent symbol before the current word is a binary operator (e.g., “+”), the current word is very likely to be a formula fragment. On this account, T-LeftAdjacent and T-RightAdjacent are defined to represent the context information.

3.3. Word classification using Support Vector Machine

To decide if a word is a formula word (or fragment) is a classical binary classification problem. In this step, SVM classifier is trained on the labeled datasets. Then, the trained classifier is utilized to predict whether a new coming word is an ordinary text word or a formula fragment.

SVM-based learning algorithm\textsuperscript{14} is well known for its generalization performance. A support vector machine can construct a hyperplane in a high dimensional space which can be considered as the optimal classification hyperplane. Let us consider a binary classification problem. The training dataset are defined as \( \{(x_1, y_1), \ldots, (x_N, y_N)\} \), where \( x_i \in \mathbb{R}^d \) is a feature vector and \( y_i \in \{-1, +1\} \) is the class label of the \( i \)-th data. If we assume that the two classes can be separated by a hyperplane \( w \cdot x + b = 0 \) in some space \( H \), the optimal hyperplane is the one which maximizes the margin \( (\|w\| = 2 / ||w||) \) between two classes. Finding the optimal hyperplane is an optimization problem to find minimum \( ||w|| \) under constraints. Furthermore, SVM can cope with the linearly inseparable training data by substituting every dot product of the features in dual form with Kernel function \( K(x_i, x_j) \).

As we discussed in Section 1, to discriminate embedded formulas from original text, one significant problem is that the rule-based quantitative models of layout features do not work well because the appropriate thresholds and parameters for classification are difficult to set. After the aforementioned steps, feature vector with 12 elements for each word has been extracted. Since SVM has excellent performance in solving classification problems in high dimensional feature space and has been successfully applied in many other pattern recognition problems, we adopt the SVM technique to classify a word into ordinary text (or formula). In our implementation, LIBSVM\textsuperscript{12}, an optimized implementation of Support Vector Machine, is used to build the SVM classifier. Radial Basis Function (RBF) is selected as the kernel function and other parameters are set by the default values in LIBSVM.

3.4. Formula region merging

When the word classification is finished, each word has a class label, namely, ordinary text word or formula fragment. The regions of the successive words classified as formula fragments are merged. In this way, embedded formula regions are finalized.

4. EXPERIMENT RESULTS

We collect the data from 50 mathematics journal papers\textsuperscript{†} and 5 mathematics textbooks\textsuperscript{§}, all in English. We randomly select 2 pages from each paper and 20 pages from each textbook. The experiments are carried on the selected 200 pages, which contain 6086 embedded formulas in total. To evaluate the performance of the SVM-based embedded formula identification method, the data set is divided into five equal subsets and five-fold cross-validation is employed. In each round, a single subset is retained as the validation data for testing the model, and the remaining four subsets are used as training data. The process is then repeated five times, with each of the subsets used exactly once as the validation data. The test results of five rounds are averaged to be the final test result.

The program is implemented in C++ and run on a 2.5GHz PC with 2GB RAM. On average, it takes 4 seconds to detect the embedded formulas on each page. It takes less than 3 second to train the SVM classifier on a training set of 160 pages. Furthermore, the proposed method has been incorporated into a commercial software package for e-Book production and about 30,000 documents have been processed by the system in the past one year.

In the following experiments, three metrics are used to evaluate the system: 1) Precision (the percentage of the extracted bounding boxes matching the formulas’ areas); 2) Recall (the percentage of that the true formulas are detected); 3) F1 is the harmonic mean of precision and recall. Comparison experiments are carried out and the results are presented as follows.

\textsuperscript{†} http://www.sciencedirect.com/science/journal/
\textsuperscript{§} http://www.springerlink.com/
We compare the performance of the SVM-based and rule-based method of embedded formula identification in Table 2. The character-rule-based method is proposed in our previous work and it extracts the embedded formulas through locating specific math symbols and applying specific context propagation from these symbols. The layout-rule-based method is implemented by constructing rules of the layout features (defined in Table 1) rather than by using classification technique. For example, if the word’s V-Fontsize is larger than the symbols’ average font size of the ordinary text words, the word is recognized as a formula fragment with a high confidence score.

### Table 2. Comparison of the performance of the rule-based and SVM-based embedded formula identification methods

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character-rule-based</td>
<td>74.45%</td>
<td>68.23%</td>
<td>71.21%</td>
</tr>
<tr>
<td>Layout-rule-based</td>
<td>78.31%</td>
<td>78.43%</td>
<td>78.37%</td>
</tr>
<tr>
<td>SVM-based</td>
<td>86.94%</td>
<td>84.29%</td>
<td>85.60%</td>
</tr>
</tbody>
</table>

An example comparing the proposed method with character-rule-based method is shown in Figure 4. In the rule-based method (a), some cases (e.g., user defined functions, unknown characters, etc.) are missed while in SVM-based method can correct them. Figure 5 is an example comparing the proposed method with layout-rule-based method. The context feature takes better effects in detecting boundary of embedded formula in SVM-based method.

![Figure 4](image1.jpg)  
(a) Character-rule-based  
(b) SVM-based

---

According to our experiments and commercial operations, errors can occur in SVM-based embedded formula identification in a number of situations: 1) Some short text lines containing embedded formulas are recognized as the isolated formulas by the upstream processing steps. Thus, embedded formulas in these lines are missed altogether. 2) The special separators (e.g., punctuation and parentheses) may be misrecognized. The main reason is that the separators, especially the parentheses, may be a part of the ordinary text or a part of the formulas with almost the same possibility. Thus, they do not stand out in terms of character features. Furthermore, as the words containing only one symbol in each word, they exhibit weak layout features. 3) In the word segmentation step, some ordinary words and adjacent embedded

![Figure 5](image2.jpg)  
(a) Layout-rule-based  
(b) SVM-based

---

An example comparing the proposed method with character-rule-based method is shown in Figure 6. In the rule-based method (a), some cases (e.g., user defined functions, unknown characters, etc.) are missed while in SVM-based method can correct them. Figure 7 is an example comparing the proposed method with layout-rule-based method. The context feature takes better effects in detecting boundary of embedded formula in SVM-based method.
formula fragments are segmented into the same word. For instance, in the expression “if $u > a$”, the ordinary word “if” and a formula fragment “$u$” are segmented in the same word. Therefore, in the following steps, this formula cannot be correctly identified.

5. CONCLUSION

In this paper, a classification-based embedded formula identification method using Support Vector Machine is introduced. The major contributions of the proposed method are as follows: 1) Machine learning technique is used for the first time to identify embedded mathematical formulas. It shows good performance and is more flexible and adaptable than the existing rule-based methods. 2) Word-specific features of embedded formulas (including geometric layout, character and context) are fully utilized. They form feature vectors to be used to train the SVM classifier. 3) A unique analytical method for embedded formula identification is proposed and it differs significantly from the existing heuristic methods. In the future, we will investigate an iterative validation process to correct the errors in our method. Mathematical formula structure recognition is also another interesting research direction.

ACKNOWLEDGMENTS

We would like to thank Shlomo Sternberg for his kindness for sharing the mathematical books on his website\textsuperscript{15}. This work is supported by National Basic Research Program of China, also named “973 Program” (No. 2010CB735908).

REFERENCES