Clustering of Handwritten Mathematical Expressions for Computer-Assisted Marking

Vu-Tran-Minh KHUONG¹, Student Member, Khanh-Minh PHAN¹, Huy-Quang UNG¹, Nonmembers, Cuong-Tuan NGUYEN¹, Member, and Masaki NAKAGAWA², Fellow

SUMMARY Many approaches enable teachers to digitalize students’ answers and mark them on the computer. However, they are still limited for supporting marking descriptive mathematical answers that can best evaluate learners’ understanding. This paper presents clustering of offline handwritten mathematical expressions (HMEs) to help teachers efficiently mark answers in the form of HMEs. In this work, we investigate a method of combining feature types from low-level directional features and multiple levels of recognition: bag-of-symbols, bag-of-relations, and bag-of-positions. Moreover, we propose a marking cost function to measure the marking effort. To show the effectiveness of our method, we used two datasets and another sampled from CROHME 2016 with synthesized patterns to prepare correct answers and incorrect answers for each question. In experiments, we employed the k-means++ algorithm for each level of features and considered their combination to produce better performance. The experiments show that the best combination of all the feature types can reduce the marking cost to about 0.6 by setting the number of answer clusters appropriately compared with the manual one-by-one marking.

key words: clustering, handwritten mathematical answers, computer-assisted marking, offline handwritten mathematical expressions

1. Introduction

Examinations, exercises, questions, and assignments play an important role in education. They help teachers evaluate students’ understanding and abilities to answer questions to prepare for the next stage of teaching. They also motivate students and guide their further learning. However, school teachers need to mark a large number of answers, which requires a huge amount of time and effort. If the feedback to the students is delayed, the learning effect is decreased[1].

To solve this problem, mark sheets and computer/web-based testing have been introduced, but they often restrict the questions to multiple-choice questions and unambiguous questions with unique answers that have been input into the system. Descriptive questions are generally excluded in such IT-empowered examinations. Although the above types of questions require less time and effort for marking, and incur fewer marking errors, they sacrifice an evaluation of the student’s deep understanding. Even if a student does not understand a question or cannot answer it, he/she can get a positive score by selecting the correct answer through guessing. It has also a negative side effect in that it encourages the students to “select” rather than “think”.

There are basically two approaches to utilize computers for marking. Automatic marking could be the ultimate solution to the exam marking problem. Extensive research has been carried out on marking essays[2]–[4] and even marking handwritten essays has been reported[5]. Several research papers have been published on auto-grading of computer programs[6]–[8], but automatic marking is generally unable to handle partially correct answers. One serious problem of automatic marking is that it cannot be used without examinees’ confirmation of marking.

A second solution is computer-assisted marking, which increases the efficiency and reliability of marking. One promising approach in this direction is to cluster answers. If answers are well clustered, they can be marked efficiently and marking errors and marking fluctuations can be reduced. Since the final markings are made by humans, the examinees’ anxieties are also reduced.

Clustering has been utilized for marking answers and assignments and this is the first step before making a marking system. Yin et al. proposed a technique for clustering student programming assignments[9]. They used abstract syntax trees to represent the syntactic structure of submitted programs and the normalized tree edit distance for clustering. For grading English short answers typed into descriptive questions, Basu et al. used k-means to group student responses into clusters and subclusters, then they automatically marked clusters and subclusters using predefined answer keywords. Teachers could read, mark, and provide feedback on these groups of answers at once[10], [11]. For handwritten answers, however, there is no published research as far as we are aware.

We focus on clustering handwritten mathematical expressions (HMEs) for computer-assisted marking, as mathematical expressions are typical descriptive answers, which reveal the students’ understanding and problem-solving abilities. Here, however, we must solve several problems to make computer-assisted marking effective. First, we must extract features from HMEs without entirely depending on their computer recognition. HME recognition is one of the most difficult handwriting recognition problems, with the best performance being around 77% for rendered images and nearly 81% on the original online patterns on the CROHME 2019 dataset[12]. Secondly, we need a measure to evaluate how “well” a clustering method supports markers in the marking process, and to compare the performance of different methods. Thirdly, we must consider the user
interaction between a marker and a marking system as clustering is not perfect. Due to many ambiguities in HMEs, the marker should be able to verify and regrade the “impure” elements efficiently. Fourthly, we must find the best clustering algorithm for this purpose. Fifthly, a lack of public datasets limits the development of research in this approach.

In this study, we make two main contributions related to the above problems. First, we present a clustering-based approach that extracts features of different types and combines them effectively for clustering offline HMEs. Secondly, we introduce a function to measure the marking cost to evaluate feature combinations and compare various marking approaches. In this paper, however, we do not consider the user interface design for marking and do not investigate clustering algorithms but use standard k-means++ because we want to focus on the first two problems mentioned above.

Our earlier research in this direction was presented as a conference paper [13]. In this paper, we reformulated that method, added a dataset composed of real and synthesized HME patterns, which is made publicly available for research purposes, and reevaluated the effect of feature combination. Particularly,

1. We reformulated the weighting combination and evaluated methods of max-min normalization and standardization for combining feature types.
2. A semi-synthetic answer dataset is prepared and made publicly available for encouraging research.
3. We reevaluated the feature types and combination methods on the semi-synthetic answer dataset.

The rest of this paper is organized as follows. Section 2 describes the feature types used in this study and their combinations. Section 3 introduces a cost function for evaluating clustering. Section 4 describes the datasets, and Sect. 5 presents the experiments and results. Finally, Sect. 6 summarizes this research and discusses possible future extensions.

2. Multi-Level Features From an HME

From an HME image, we can extract low-level and high-level features without using a complete HME recognizer to generate a tree or a LaTeX notation, where low-level features represent simple features such as edges and corners extracted by various filters, and high-level features represent abstract features extracted by symbol segmentation and recognition methods. Figure 1 shows these features according to their levels. The low-level features are robustly extracted from an HME image but they are often sensitive to the way it is written.

As for the high-level features, we extract symbols, recognize them, represent their positions and relations in the form of bag-of-symbols, bag-of-relations, and bag-of-positions, without applying the HME recognition yet. This is because the HME recognition applies structural analysis and total optimization in the context of a mathematical expression grammar on top of symbol segmentation and recognition, so that the entire process is often fragile [12], [14]–[18]. Therefore, we avoid the last stage of structural analysis and total optimization. A convolutional neural network (CNN) is applied for recognizing symbols as detailed in Sect. 2.2. These bag-of-features are considered as high-level features as they are extracted from symbols recognized from an HME. The high-level features are dependent on symbol recognition but are robust with respect to the way an HME is written in different styles.

In the experiments, we evaluate the effect of the low-level and high-level features and consider combining them to improve the clustering performance. The rest of this section describes these features in detail and present the combination strategies.

2.1 Directional Features

Directional features have been used since the 1980s for handwritten character recognition [19], [20]. We extract directional features from an HME image in four steps: nonlinear normalization, directional decomposition, Gaussian blurring, and assembling features into a single vector as shown in Fig. 2. First, an HME image is normalized to a fixed size by nonlinear normalization [21]. Secondly, we extract edges in the normalized image, project them into 8-direction planes, with each plane containing one direction only. Thirdly, we divide each plane into a predefined mesh of cells and apply a low-pass Gaussian filter for all the cells...
with some overlap to enhance the robustness to positional distortions. Finally, we concatenate all the directional magnitudes of all the cells and all the planes into a feature vector.

In this work, we choose the normalized size in the first step as the average height and width of all input HME images, denoted as $\bar{H}$ and $\bar{W}$, respectively. Then, the mesh of cells is defined as $R \times C$. To effectively capture the directional feature, the mesh is defined so that each cell can capture at most one symbol. Given a set of normalized HME images $\{I_1, I_2, \ldots, I_N\}$, we calculate the average height and width of the connected components within the HME image $I_i$, denoted as $\bar{H}^{h}_{CC}$ and $\bar{W}^{h}_{CC}$, respectively. Then, we compute the maximum among $\bar{H}^{h}_{CC}$ such that $\bar{H}_{comp} = \max(\bar{H}^{h1}_{CC}, \bar{H}^{h2}_{CC}, \ldots, \bar{H}^{hN}_{CC})$ and the maximum among $\bar{W}^{h}_{CC}$ such that $\bar{W}_{comp} = \max(\bar{W}^{h1}_{CC}, \bar{W}^{h2}_{CC}, \ldots, \bar{W}^{hN}_{CC})$. Dividing $\bar{H}$ and $\bar{W}$ by $\bar{H}^{h}_{CC}$ and $\bar{W}^{h}_{CC}$, respectively as shown in Eq. (1), we obtain the mesh of $R \times C$ so that each cell contains at most one symbol.

$$R = \frac{\bar{H}}{\bar{H}_{comp}}, \quad C = \frac{\bar{W}}{\bar{W}_{comp}}$$

Therefore, the size of the directional feature vector is $8 \times R \times C$.

### 2.2 Bag-of-Symbols

Bag-of-symbols is often used for clustering. Symbols in HMEs are digits, letters, operators, several kinds of brackets, etc. Hence, bag-of-symbols is an S-dimensional vector, where $S$ is the number of symbol categories. Instead of using the occurrence counts of symbols, however, we use their occurrence probabilities because these give more information when misrecognitions occur and can be retrieved from the symbol recognition. We employ a CNN architecture presented in [22]. This CNN is applied for each connected component in an HME image to obtain the vector of occurrence probabilities of the symbol categories. Then, we add all the vectors to form the feature vector of the entire HME image by Eq. (2).

$$BoS = \sum_{i=1}^{nCC} BoS_{CC,i}$$

where $BoS$ is the feature vector of bag-of-symbols from the entire HME image, $BoS_{CC,i}$ is that of bag-of-symbols from the $i$th connected component, and $nCC$ is the total number of connected components inside the HME image.

For symbols composed of multiple components, a set of rules is predefined for combining them as shown in Table 1. Each rule is associated with a threshold to reduce wrong combinations. According to the rules, if some individual components are combined and reclassified as the corresponding symbol with a higher probability than the corresponding threshold by the CNN, they are recognized as the corresponding symbol. For example, if a pair of "-" components is reclassified as the "=" symbol by the CNN with a probability higher than 0.7, these "-" components are recognized as the "=" symbol.

### 2.3 Bag-of-Relations

The relationship between the components under consideration is represented by their relative position to each other. A bag-of-relations represents how many types of relations occur in an HME. The bag-of-relations of each component counts its frequencies of relations with the neighbor components. Figure 3 describes the bag-of-relations. Figure 3 (a) shows the two-dimensional space around the component under consideration, which is divided into nine regions and numbered as shown in Fig. 3 (a). Each region represents a specific relation between the components as shown in Fig. 3 (b). For each extracted component, its neighboring components and the relationships among them are determined by xy-projections. Two components are neighbors to each other if there is no other component whose bounding box overlaps the line connecting the centers of their bounding boxes. Then, we count the number of connected components in each region based on the centers of their bounding boxes so that this feature value is discrete. Figure 3 (c) shows an example of extracting a bag-of-relations for the component "a". For each component, we produce a feature vector for bag-of-relations. After extracting the bag-of-relations from all the components, we calculate their cumulative sum for forming the final feature vector of the entire HME pattern.

### Table 1 Rules for combining connected components.

| Component 1 | Component 2 | Symbol | Threshold |
|-------------|-------------|--------|-----------|
| -           | -           | =      | 0.7       |
| <           | -           | ≤      | 0.8       |
| >           | -           | ≥      | 0.8       |
| -           | -           | ?      | 0.5       |
| +           | -           | ±      | 0.5       |

![Fig. 3 Bag-of-relations.](image)
2.4 Bag-of-Positions

A bag-of-positions represents where components exist in an HME image. We extract the bounding box of an input HME and divide it into uniform bins of rectangles (M×N bins). Hence, a feature vector for bag-of-positions is an MN-dimensional vector.

In this work, the number of bins is chosen based on the structure of the true answer assuming that students write the true answer and wrong answers similar to the true answer. When students write an answer, they usually imagine a baseline to guide the following writing. Moreover, an HME often has relations of superscript, subscript, above and below, which leads to creating sub-baselines for writing terms having these relations. The number of sub-baselines of an expression $SB_{Exp}$ are determined by Eq. (3):

$$SB_{Exp} = \max (SB_{sup}, SB_{above}) + \max (SB_{sub}, SB_{below})$$  \hspace{1cm} (3)

where $SB_{sup}$, $SB_{sub}$, $SB_{above}$, $SB_{below}$ are the number of sub-baselines of the superscript term, that of the subscript term, that of the above term and that of the below term, respectively. We assume that the terms at the same level in a nested structure share the same sub-baseline. For example, in the expression “$a^{b}$”, “$a^{c}$” and “$a^{f}$” share the same sub-baseline.

The number of rows and columns of bins are determined by examining the baseline and sub-baselines in the answer. The number of rows is chosen to be the total number of the baseline and sub-baselines, while the number of columns is the number of symbols on the baseline plus the largest number of symbols among the sub-baselines. Figure 4 shows an example. The number of rows is three as there is one baseline and two sub-baselines for the numerator and the denominator of the fraction. The number of columns is five as there are two symbols on the baseline and three symbols (the largest number of symbols) on the sub-baselines.

For each component, we compute the distribution of the component’s pixels in each bin by Eq. (4):

$$P(G, C) = \frac{\# \text{ pixels of } C \text{ inside } G}{\# \text{ pixels of component } C}$$  \hspace{1cm} (4)

Then, we apply a Gaussian Filter with $\sigma_x$ and $\sigma_y$ being $\frac{\sqrt{\pi}}{2} \times g_x$ and $\frac{\sqrt{\pi}}{2} \times g_y$ where $g_x$ and $g_y$ are the width and height of each bin. The final feature vector is the cumulative summation of all the components by Eq. (5):

$$P = \left\{ \sum_C P(G_{11}, C), \sum_C P(G_{12}, C), \ldots, \sum_C P(G_{NM}, C) \right\}$$  \hspace{1cm} (5)

where $G_{nm}$ (1 ≤ n ≤ N, 1 ≤ m ≤ M) is the bin at the nth row and the mth column.

2.5 Feature Combination Methods

Combining multiple feature types is expected to form more robust features for clustering HMEs. Eq. (6) presents the concatenated form of these feature types:

$$F = (F_1, F_2, \ldots, F_i, \ldots, F_H)$$  \hspace{1cm} (6)

where $F_i = (f_{i1}, f_{i2}, \ldots, f_{iD_i})$ (1 ≤ i ≤ H) and $D_i$ is the dimension of $F_i$.

Given two data points $p$, $q$ in the dataset $Dset$, the distance between their combined features $F^p$ and $F^q$ is the cumulative distance over each feature type $F^p_i$, $F^q_i$ as shown in Eq. (7):

$$d(F^p_i, F^q_i) = \sum_{i=1}^{K} d(F^p_i, F^q_i)$$  \hspace{1cm} (7)

where $d(F^p_i, F^q_i)$ is the distance between the feature types $F^p_i$ and $F^q_i$. We used Euclidean distance.

Then, we present three methods to combine feature types for clustering, namely normalization, standardization, and weighted combination. The first method, denoted as Normalized_Comb, normalizes the individual feature vectors and concatenates them together. The normalized form of $F$ is shown in Eq. (8):

$$\hat{F} = (\hat{F}_1, \hat{F}_2, \ldots, \hat{F}_i, \ldots, \hat{F}_H)$$  \hspace{1cm} (8)

where $\hat{F}_i$ is the normalized form of $F_i$ by scaling the values of each feature type to the interval of [0, 1]. Given a data point $p$ in $Dset$, each individual feature type $F^p_i$ (1 ≤ i ≤ K) is normalized as shown in Eq. (9):

$$\hat{F}^p_i = \frac{F^p_i - \min_{\tilde{f}^p_i \in F^p_i, \tilde{f}^p_i \in Dset} \tilde{f}^p_i}{\max_{\tilde{f}^p_i \in F^p_i, \tilde{f}^p_i \in Dset} \tilde{f}^p_i - \min_{\tilde{f}^p_i \in F^p_i, \tilde{f}^p_i \in Dset} \tilde{f}^p_i}$$  \hspace{1cm} (9)

The second method, denoted as Standardized_Comb, makes the distribution of each feature type have zero-mean and unit variance. Given a data point $p$ in $Dset$, each individual feature type $F^p_i$ (1 ≤ i ≤ K) is standardized as shown in Eq. (10):

$$\hat{F}^p_i = \left( \frac{f^p_i - \mu_{i1}}{\sigma_{i1}}, \frac{f^p_i - \mu_{i2}}{\sigma_{i2}}, \ldots, \frac{f^p_i - \mu_{iD_i}}{\sigma_{iD_i}} \right)$$  \hspace{1cm} (10)

where $\mu_{ij}$ is the mean value and $\sigma_{ij}$ is the standard deviation of $f_{ij}$.

The third method, denoted as Weighted_Comb, uses a set of weighting parameters for all the feature types to optimize clustering performance. As the clustering performance of each feature type is different, the Weighted_Comb allows feature types to contribute to the clustering at different degrees using the weighting parameters. The feature type
which contributes more effectively is weighted higher than the less effective one. A training set is used to determine the optimal weighting parameters. We use the grid-search technique to find the optimal weighting parameters.

The weighted distance is calculated by extending Eq. (7) as shown in Eq. (11):

\[
d(F^p, F^q) = \sum_{i=1}^{K} w_i d(F^p_i, F^q_i)
\]

where \(w_i\) is the weight trained for the distance in the feature type \(F_i\) with \(\sum_i w_i = 1\).

2.6 Positional Bag-of-Features

Another approach for improving the clustering performance is extracting bag-of-features by local regions. We call it positional bag-of-features. Particularly, we divide the image into a mesh as the bag-of-positions feature. For each bin in the mesh, we extract bag-of-symbols and bag-of-relations. Then, we concatenate these bag-of-features bin by bin. The advantage of this approach is preserving the information of relative positions among symbols.

3. Marking Cost Function

In this study, we propose a marking cost function for evaluating clustering-based marking assistance without requiring data from human participants.

Assume that \(W\) is a set of clusters \(\{w_1, w_2, \ldots, w_K\}\), with \(K\) being the number of clusters, and \(C\) is a set of classes \(\{c_1, c_2, \ldots, c_J\}\), with \(J\) being the number of classes. For each cluster \(w_k\) in \(W\), let \(M_k\) be the set of samples belonging to the major class \(c_j\) (the class \(c_j\) has the most samples) in the cluster \(w_k\) as shown in Eq. (12):

\[
|M_k| = \max_{i \leq j \leq J} \left| \{w_k \cap c_j\} \right|
\]

Then, the purity is the accuracy of this assignment on the whole dataset divided by the total number of samples \(N\), as shown in Eq. (13):

\[
Purity(W, C) = \frac{1}{N} \sum_{k=1}^{K} |M_k|
\]

High purity is easy to achieve when the number of clusters is huge. For example, if \(K\) equals to \(N\), we obtain the purity of one. Thus, the purity itself does not show the clustering quality.

We propose a cost function to evaluate the contribution of a marker during the marking process. Our cost function addresses the deficiency of the purity score by considering both the purity and the number of clusters. Given a cluster of answers, if there exist more than two different sets of answers, the marker tends to weed out the minor answers and retain the major one. Thus, the major answers just need a single marking operation while the minor ones require the same amount of effort as the manual approach.

Marking may depend on the size (or length) of answers and the nature of questions. However, we neglect these issues as the first step by considering a simple scenario of verifying and marking answers, where the task of verifying is to find different answers inside a cluster and the task of marking is to compare an answer with the correct answer. Both the tasks need to detect differences between an answer with other(s). Though detecting differences may depend on the sizes of expressions, we can take their average and assume little dependence on the question except for specific cases.

Assuming that a marking cost and a verification cost (in terms of time) are incurred per answer. Let \(T\) be the average marking cost of all answers. The verification cost is expected to be lower than the marking cost. We assume that there exist a real number \(\alpha(0 < \alpha \leq 1)\) so that the average verification cost is \(\alpha T\). Considering a cluster \(w_i\), which is separated into a set of major answers \(M_i\) and some sets of minor answers, its marking cost is calculated as shown in Eq. (14):

\[
\text{cost}(w_i, C) = C_{\text{marking}} + C_{\text{verification}} = (1 + |w_i| - |M_i|) \times T + \alpha |w_i| \times T
\]

where \(C_{\text{marking}} = (1 + |w_i| - |M_i|) \times T\) is the cost of marking the single major set and \(|w_i| - |M_i|\) samples in the minor sets, and \(C_{\text{verification}} = \alpha |w_i| \times T\) is the cost of verifying all the answers in the cluster \(w_i\).

Thus, the cost of the whole dataset is given in Eq. (15):

\[
\text{cost}(W, C) = \sum_{w_i \in W} \text{cost}(w_i, C)
\]

\[
= \left( K + N - \sum_i |M_i| \right) \times T + \alpha NT
\]

We rewrite Eq. (15) using the purity term as shown in Eq. (16):

\[
\text{cost}(W, C) = \sum_{w_i \in W} \text{cost}(w_i, C)
\]

\[
= \left( K + (1 - \text{Purity}(W, C)) \times N \right) \times T + \alpha NT
\]

To estimate the worst case, we assume that the verification cost is as large as the marking cost (\(\alpha = 1\)) and rewrite the cost function in Eq. (16) as shown in Eq. (17):

\[
\text{cost}(W, C) = (K + (1 - \text{Purity}(W, C)) \times N) \times T + NT
\]

Since the maximum value of cost \((W, C)\) in Eq. (17) is \(2NT\), we normalize the value range of Eq. (17) into the interval of \([0, 1]\) by dividing it by \(2NT\). Our final cost function is shown in Eq. (18):

\[
f(W, C) = \frac{K}{2N} + \left( 1 - \frac{1}{2}\text{Purity}(W, C) \right)
\]

Note that \(f\) approaches 1 in the worst case, implying that the marking cost approaches the manual one-by-one
marking cost. In the HME clustering problem, we try to minimize this cost function with respect to the number of clusters K and the purity. Note that this cost function indicates the worst case since we assumed $\alpha = 1$ and cost $(W, C) = 2NT$ as shown in Eq. (17). By designing an effective user interface for makers, the value of $\alpha$ can be decreased and the marking cost is reduced.

### 4. Datasets

We use three datasets of offline HME patterns to evaluate the features and methods. Here, offline HME patterns are made from online HME patterns by connecting adjacent points with straight lines and thickening them with a constant width of five pixels. These datasets share the same 101 Math symbol categories mentioned in [15], [17]. The first dataset named “Dset_22Qs” was collected from 23 students (19 males and 4 females). Each student wrote three prepared answers: a correct answer and two incorrect (partially incorrect and totally incorrect) answers for each of the 22 questions, resulting in a total number of samples: $23 \times 3 \times 22 = 1,518$.

The second dataset named “Dset_50” was collected from 21 students to evaluate the performance with a larger number of different answers. This data set was originally collected to test HME recognition on three different user interfaces: (1) without any guiding line; (2) with a center line; and (3) with center, top, and bottom lines. Each student wrote 50 different HMEs on the three interfaces, thereby generating $21 \times 50 \times 3 = 3,150$ HME patterns. Although its original purpose was different, Dset_50 can be used for evaluating clustering of HMEs. The details of these datasets are shown in Table 2.

The third dataset was sampled from the CROHME 2016 dataset [14], but it was not enough so it was augmented by synthetic patterns from LaTeX sequences and isolated handwritten symbol patterns in CROHME 2016. This augmented data set is named “Dset_Mix”, and is organized so that real (handwritten) HME patterns and synthetic patterns are mixed in a way that each question has a few correct answers and several incorrect answers. Although real answers are the best, collecting correct and incorrect answers requires data from real examinations with participants’ agreements. Therefore, we took the synthetic approach to augment real HME patterns like [23] and made the dataset available for public use. We prepared 200 answers for each of 10 questions, which included handwritten and synthetic answers, as well as correct and incorrect answers. This sample size is set according to the number of students in each grade in a typical school. We published the dataset along with the list of the 10 questions.

Figure 5 shows the process of creating a synthesized pattern using the tool described in [24]. Given a LaTeX sequence, the tool first prepares a template by reflecting the sizes and positions of the symbols based on their relations to each other (horizontal, superscript, subscript, above, below, inside). Second, it translates bounding boxes in the template randomly, Third, it resizes and fills handwritten pat-

![Fig. 5 Process to synthesize an HME.](http://tc11.cvc.uab.es/datasets/Dset_Mix.html)

\[
\begin{align*}
\text{correct answer (handwritten)} & \quad \frac{-b + \sqrt{b^2 - 4ac}}{2a} \\
\text{incorrect answer (handwritten)} & \quad \frac{-b - \sqrt{b^2 - 4ac}}{2a} \\
\text{incorrect answer (synthetic)} & \quad \frac{-b + \sqrt{b^2 - 4ac}}{2a}
\end{align*}
\]

![Fig. 6 Samples in Dset_Mix.](http://tc11.cvc.uab.es/datasets/Dset_Mix_1.html)

| Subgroup No. | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|--------------|---|---|---|---|---|---|---|---|---|----|
| # categories of correct answers | 2 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 |
| # handwritten patterns of correct answers | 26 | 0 | 0 | 20 | 20 | 20 | 20 | 27 | 0 | 0 |
| # synthetic patterns of correct answers | 21 | 40 | 50 | 20 | 35 | 10 | 81 | 49 | 50 | 50 |
| # categories of incorrect answers | 8 | 4 | 5 | 4 | 4 | 4 | 6 | 2 | 5 | 3 |
| # handwritten patterns of incorrect answers | 21 | 18 | 3 | 19 | 2 | 39 | 1 | 27 | 0 | 0 |
| # synthetic patterns of incorrect answers | 132 | 142 | 147 | 141 | 143 | 131 | 98 | 97 | 150 | 150 |
| # total answers | 200 | 200 | 200 | 200 | 200 | 200 | 200 | 200 | 200 | 200 |
terns into the prepared template. Finally, symbol patterns from CROHME 2016 are filled into the template.

Dset Mix is divided into ten subgroups corresponding to ten questions. Each subgroup stores one or two correct answers and one to five incorrect ones with each having one or multiple HME patterns. Table 3 shows the number of HME categories and patterns in each subgroup. Figure 6 shows some sample patterns in subgroup 1 for the question 1. According to Table 3, Dset Mix consists of 2,000 HMEs (263 handwritten HMEs from CROHME 2016 and 1,737 synthetic patterns).

5. Experiments and Results

In this section, we seek to find the best feature type or feature combination and the best combination method.

5.1 Experiments on Individual Feature Types

The number of symbol categories in the datasets is 101 so that the dimension of the bag-of-symbols feature vector is 101 ($k = 101$). We trained a CNN on the isolated symbols in CROHME 2013 and achieved a symbol recognition rate of 89.39% on CROHME 2014.

The clustering performance of the feature types was evaluated individually. For Dset 22Qs, we applied the clustering method question by question (the number of classes is 3 for each question). For Dset 50, we applied the clustering method on the whole dataset so that the number of classes was 50. For Dset Mix, we employed the clustering method question by question (the number of classes varied from 4 to 10). We used $k$-means++. For each type, we fixed $k$ (the only parameter of $k$-means++) to be equal to the number of classes for evaluating its discrimination ability, although the number of true classes is unknown for real situations. The Euclidean distance was used for clustering.

Table 4 shows the purity by the individual feature types for Dset 22Qs, Dset 50, and Dset Mix in terms of the average purity for the 22 questions in Dset 22Qs, the purity for 50 ME categories in Dset 50 and the average purity for ten sets in Dset Mix. The results show that the bag-of-symbols feature achieves the highest purity among the individual feature types on the three datasets. Moreover, Dset Mix shows similar purity levels as Dset 22Qs and Dset 50 for all the feature types, which means that it may be used for further evaluation.

Positional BoS, Positional BoR and their combination are inferior to simple Bag-of-symbols, except their combination for Dset Mix. The reason seems that concatenating the bag-of-features of all the bins makes their dimension huge and the features sparse as known as the curse of dimensionality. Hence, the clustering performance is not improved.

5.2 Experiments on Feature Combination

We evaluated the methods of combining individual feature types. To evaluate Weighted Comb, we needed to train its weights, although we did not need to train Normalized Comb and Standardize Comb. Therefore, we applied 5-fold cross-validation for Weighted Comb. For the cross-validation, one subset was reserved for testing, and the remaining four subsets were used for determining the optimal set of weights by employing the grid search technique and measuring the performance on the testing subset in each round. The average across all the five testing subsets was taken. The entire Dset 50 was used for training to obtain the optimal weights, which were applied for Weighted Comb on Dset 22Qs and Dset Mix. For Normalized Comb and Standardize Comb, we employed these methods directly.

There can be two different ways to form the five folds. One way is to split the dataset by the HME categories, and the other is to split it by the writers. The former evaluates the clustering performance on “unseen” HME categories, whereas the latter evaluates the clustering performance on “unseen” writing styles. In this study, we take the former approach and split the dataset into five subsets by the HME categories as a proper clustering-based marking assistance should be able to cluster unseen HME categories.

Table 5 shows the mean purity and standard deviation for combining various feature types by each combination method presented in Sect. 2.5. As the bag-of-symbols is the best single feature type, we examined its combinations with others. We show the results in detail for Weighted Comb, but the other results are similar so that they are partially omitted. When this feature is combined with another feature type, the best results are achieved with the directional feature. When combined with two other feature types, the effect is small or even negative. When combined with the three other feature types, the effect is maximized. In this case, the weighting parameters of BoS, Dir, BoR, BoP are 0.45, 0.35, 0.15, 0.05, respectively. As we can see, BoS and Dir whose clustering performance is high are assigned with higher weights while BoR and BoP are assigned with lower weights.

As Weighted Comb learns the significance of each feature type through training samples, it performs better than Normalized Comb and Standardized Comb which normalizes or equalizes the role of each feature type. We also considered applying normalization and standardization before employing Weighted Comb after the feature normalization. The performance is significantly reduced with the purities are around 0.7485 (normalization) and 0.7400 (standardization). Applying normalization or standardization may scale down the magnitudes of the features and make the
distances in the feature space of BoS and Dir similar to BoR and BoP. Hence, the grid-search method hardly finds the most significant features with the reduced performance. For example, after applying the standardization, the grid search method yielded the weight (w_{dir}, w_{bos}, w_{bor}, w_{bop}) = (0.2, 0.1, 0.5, 0.2) in which BoR was assigned the largest weight though the purity of this feature alone was not as good as BoS.

5.3 Experiments on Marking Cost for Dset_Mix

To investigate the marking cost, we evaluated it on Dset_Mix as Dset_Mix has 200 answers for 10 questions each, including correct and incorrect answers. It is expected to simulate the marking of handwritten math answers in schools. We applied the k-means clustering using the best feature combination method, i.e., Weighted_Comb, for question by question. Although the true number of classes among the answers for each question is from 4 to 10 from Table 3, we can not know this when marking the answers. Therefore, we considered the purity and the marking cost according to the parameter k of the k-means method. Figure 7 shows the average purity and the marking cost of the 10 questions along the logarithmic scale for the parameter k.

When k increases up to 12, the purity increases and the marking cost decreases. In this case, an appropriate number of clusters helps the k-means algorithm adapt to the data distribution and the marking cost decreases. When the number of clusters is 12, the marking cost achieves the minimum value (0.62). When the number of clusters further goes up over 12, the purity also continues to increase and approaches 1 but the marking cost starts to increase and approaches 1 eventually. As we expected, too many clusters make the k-means algorithm produce more groups of answers to be marked so that the marking cost increases. This result shows that the marking cost is determined as a trade-off between the clustering quality and the number of clusters.

The result also shows that the marking cost is rather stable between k = 8 and 32 so that we can set the number of k as such, provided that the true number of classes are

---

**Table 5** Purities of individual and combined features (mean ± standard deviation).

| Subgroup No. | Feature types | Dset_50 (5-fold cross-validation) | datasets |
|--------------|---------------|-----------------------------------|----------|
|              |               | BoS                  | Dir                  | BoR                  | BoP                  |
| Normalized_Comb | ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ | 0.8518 ± 0.1174 | 0.7182 ± 0.1032 | 0.7405 ± 0.1838 |
| Standardized_Comb | ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ | 0.8651 ± 0.1120 | 0.6856 ± 0.1058 | 0.7555 ± 0.1705 |
| Weighted_Comb | ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ | 0.8635 ± 0.1138 | 0.6954 ± 0.1049 | 0.7580 ± 0.1608 |
| ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ | 0.8786 ± 0.1149 | 0.7186 ± 0.1040 | 0.7835 ± 0.1645 |
| Normalized + Weighted_Comb | ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ | 0.8601 ± 0.1281 | 0.7540 ± 0.1039 | 0.7410 ± 0.1637 |
| Standardized + Weighted_Comb | ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ ✓ | 0.8610 ± 0.1215 | 0.7557 ± 0.1033 | 0.7580 ± 0.1803 |
|              |               | 0.8705 ± 0.1250 | 0.7650 ± 0.1141 | 0.7585 ± 0.1692 |
|              |               | 0.8737 ± 0.1179 | 0.7775 ± 0.1170 | 0.7530 ± 0.1735 |
|              |               | 0.8514 ± 0.1263 | 0.7642 ± 0.1779 | 0.7864 ± 0.0746 |
|              |               | 0.9270 ± 0.1240 | 0.7868 ± 0.1121 | 0.8285 ± 0.1648 |
|              |               | 0.8614 ± 0.1786 | 0.7745 ± 0.1167 | 0.7570 ± 0.0846 |
|              |               | 0.8771 ± 0.1262 | 0.7751 ± 0.1105 | 0.7870 ± 0.1113 |
|              |               | 0.9167 ± 0.1163 | 0.7835 ± 0.1161 | 0.8045 ± 0.1667 |
|              |               | 0.8700 ± 0.1287 | 0.7768 ± 0.1176 | 0.7879 ± 0.1134 |
|              |               | 0.9206 ± 0.1182 | 0.7774 ± 0.1203 | 0.8175 ± 0.1576 |
|              |               | **0.9303 ± 0.1189** | **0.7943 ± 0.1061** | **0.8410 ± 0.1515** |
|              |               | 0.8113 ± 0.1839 | 0.7081 ± 0.1677 | 0.7485 ± 0.1658 |
|              |               | 0.8258 ± 0.1516 | 0.7468 ± 0.1774 | 0.7400 ± 0.1876 |

---

**Fig. 7** Purity and marking cost on Dset_Mix.

**Fig. 8** Purity and marking cost on question 1: If a quadratic function \( ax^2 + bx + c = 0 \) has two roots, what is the general form of the roots? (the true number of answer classes is 10).
We presented the clustering of offline handwritten mathematical expressions (HMEs) to help teachers efficiently mark answers in the form of HMEs. We investigated multiple levels of features from HMEs, i.e., directional features, bag-of-symbols, bag-of-relations, bag-of-positions, and their combinations. We also proposed a cost function to reflect the marking effort. We conducted experiments using the three datasets, Dset_22Qs, Dset_50 and Dset_Mix which was prepared to simulate an examination composed of 10 questions for 200 students. The results shows: the bag-of-symbols yields the best result among individual features; the combination of all the feature types by the trained Weighted_Comb method achieves the best result of reducing the marking cost to about 0.6.

There remain some research issues. First, true handwritten answers must be collected from real examinations. Secondly, the clustering method must be elaborated for the marking purpose. We just employed the most basic clustering method of $k$-means, which requires us to set the number of clusters beforehand. The number of clusters must be roughly set before applying our clustering by evaluating the proposed cost function for a range of the clustering number on a training dataset. However, we are investigating more advanced estimating methods as well as clustering algorithms that do not require specifying the number of clusters. Thirdly, a marking system must be prototyped to elaborate on the marking cost function. Finally, complete offline HME recognizers trained by the latest learning method must be tested to extract features and compared with our proposed method.

6. Conclusions

We presented the clustering of offline handwritten mathematical expressions (HMEs) to help teachers efficiently mark answers in the form of HMEs. We investigated multiple levels of features from HMEs, i.e., directional features, bag-of-symbols, bag-of-relations, bag-of-positions, and their combinations. We also proposed a cost function from 4 to 10 for 200 answers. Here, the Stable Region of a parameter $p$ with an objective function $f$ is the interval of the parameter in which the difference of the maximum and minimum values of the objective function $f$ does not exceed a given threshold $\epsilon > 0$.

Figure 8 and Fig. 9 show the cases for the question 1, where the true number of answer classes is the largest (10), and question 7, where that is the smallest (4). For both these cases, the marking cost is stable between $k = 8$ and 32 since the marking cost does not vary more than 0.1 ($\epsilon=0.1$) in this interval and it is reduced to about 0.6. This implies that the value of $k$ is not so sensitive to the results for all the questions. Figure 10 visualizes the clustering result of the answers for question 1 by applying t-SNE.

References

[1] M. Thorpe, “Assessment for retention and learning: Design, feedback and quality,” Int. Symposium 2012 Student Assessment in Distance Learning and e-Learning, pp.15–23, 2012.
[2] R. Cummins, M. Zhang, and T. Briscoe, “Constrained multi-task learning for automated essay scoring,” Proc. 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Berlin, Germany, pp.789–799, 2016.
[3] S. Valenti, F. Neri, and A. Cucchiarelli, “An overview of current research on automated essay grading,” J. Information Technology Education, vol.2, no.1, pp.319–330, 2003.
[4] T. Ishioka and M. Kameda, “Automated japanese essay scoring system: jess,” Proc. 15th Int. Workshop on Database and Expert Systems Applications, Zaragoza, Spain, pp.4–8, IEEE, 2004.
[5] S. Srihari, J. Collins, R. Srihari, H. Srinivasan, S. Shetty, and J. Brutt-Griffler, “Automatic scoring of short handwritten essays in reading comprehension tests,” Artifi. Intell., vol.172, no.2-3, pp.300–324, Feb. 2008.
[6] E.L. Glassman, J. Scott, R. Singh, P.J. Guo, and R.C. Miller, “Overcode: Visualizing variation in student solutions to programming problems at scale,” ACM Trans. Computer-Human Interaction (TOCHI), vol.22, no.2, pp.1–35, March 2015.
[7] D.J. Malan, “CSS5 sandbox: secure execution of untrusted code,” Proc. 44th ACM technical symposium on Computer science education, Colorado, USA, pp.141–146, March 2013.
[8] M.T. Helmick, “Interface-based programming assignments and automatic grading of java programs,” Proc. 12th annual SIGCSE Conf.
on Innovation and technology in computer science education, pp.63–67, June 2007.

[9] H. Yin, J. Moghadam, and A. Fox, “Clustering student programming assignments to multiply instructor leverage,” Proc. 2nd ACM Conf. on Learning@ Scale, pp.367–372, March 2015.

[10] S. Basu, C. Jacobs, and L. Vanderwende, “Powergrading: a clustering approach to amplify human effort for short answer grading,” Trans. of the Association for Computational Linguistics, vol.1, pp.391–402, 2013.

[11] M. Brooks, S. Basu, C. Jacobs, and L. Vanderwende, “Divide and correct: using clusters to grade short answers at scale,” Proc. 1st ACM Conf. Learning@ scale, pp.89–98, March 2014.

[12] M. Mahdavi, R. Zanibbi, H. Mouchere, C. Viard-Gaudin, and U. Garain, “ICDAR 2019 CROHME + TFD: Competition on recognition of handwritten mathematical expressions and typeset formula detection,” Proc. 15th Int. Conf. Document Analysis and Recognition, Sydney, Australia, pp.922–927, Sept. 2019.

[13] V.T.M. Khuong, H.Q. Ung, C.T. Nguyen, and M. Nakagawa, “Clustering online handwritten mathematical answers for computer-assisted marking,” Proc. 1st Int. Conf. on Pattern Recognit. and Artificial Intelligence, Montreal, Canada, pp.121–126, May 2018.

[14] H. Mouchere, C. Viard-Gaudin, R. Zanibbi, and U. Garain, “ICFHR 2016 CROHME: Competition on recognition of online handwritten mathematical expressions,” Proc. 15th Int. Conf. on Frontiers in Handwriting Recognition, Shenzhen, China, pp.607–612, IEEE, Oct. 2016.

[15] H. Mouchere, C. Viard-Gaudin, D.H. Kim, J.H. Kim, and U. Garain, “CROHME 2011: Competition on recognition of online handwritten mathematical expressions,” Proc. 11th Int. Conf. on Document Analysis and Recognition, Beijing, China, pp.1497–1500, IEEE, Sept. 2011.

[16] H. Mouchere, C. Viard-Gaudin, D.H. Kim, J.H. Kim, and U. Garain, “ICFHR 2012 Competition on recognition of on-line mathematical expressions (CROHME 2012),” Proc. 13th Int. Conf. on Frontiers in Handwriting Recognition, Bari, Italy, pp.811–816, IEEE, Sept. 2012.

[17] H. Mouchere, C. Viard-Gaudin, R. Zanibbi, U. Garain, D.H. Kim, and J.H. Kim, “ICDAR 2013 CROHME: Third international competition on recognition of online handwritten mathematical expressions,” Proc. 12th Int. Conf. on Document Analysis and Recognition, Washington, USA, pp.1428–1432, IEEE, Aug. 2013.

[18] H. Mouchere, R. Zanibbi, U. Garain, and C. Viard-Gaudin, “Advancing the state of the art for handwritten math recognition: the CROHME competitions, 2011–2014,” Int. J. Document Analysis and Recognition, vol.19, no.2, pp.173–189, 2016.

[19] S. Mori, K. Yamamoto, and M. Yasuda, “Research on machine recognition of handprinted characters,” IEEE Trans. Pattern Anal. Mach. Intell., vol.PAMI-6, no.4, pp.386–405, July 1984.

[20] S. Mori, C.Y. Suen, and K. Yamamoto, “Historical review of OCR research and development,” IEEE, vol.80, no.7, pp.1029–1058, July 1992.

[21] C.L. Liu and K. Marukawa, “Pseudo two-dimensional shape normalization methods for handwritten chinese character recognition,” Pattern Recognit., vol.38, no.12, pp.2242–2255, Dec. 2005.

[22] H. Dai Nguyen, A.D. Le, and M. Nakagawa, “Deep neural networks for recognizing online handwritten mathematical symbols,” Proc. 3rd IAPR Asian Conf. on Pattern Recognit., Kuala Lumpur, Malaysia, pp.121–125, IEEE, Nov. 2015.

[23] S.I. Nikolenko, “Synthetic data for deep learning,” arXiv preprint arXiv:1909.11512, 2019.

[24] V.T.M. Khuong, U.Q. Huy, N. Masaki, and M.K. Phan, “Generating synthetic handwritten mathematical expressions from a latex sequence or a mathml script,” Proc. 15th Int. Conf. on Document Analysis and Recognition, Sydney, Australia, pp.922–927, IEEE, Sept. 2019.

Vu-Tran-Minh Khuong was born on 21 Jan. 1991. He received his B.Sc. degree from Ho Chi Minh Univ. of Sciences in 2014 and his M.Sc. from Tokyo Univ. of Agriculture and Technology (TUAT) in 2018. He is currently a Ph.D. candidate at TUAT and his research is on computer-assisted marking of handwritten mathematical answers.

Khanh-Minh Phan was born on 7 Aug. 1990. He received his M.Sc and Ph.D. degrees from TUAT in 2016 and 2019, respectively. Since May 2019, he is an Assistant Professor at TUAT. He is currently working on handwritten mathematical expression recognition and computer-assisted marking of handwritten mathematical answers.

Huy-Quang Ung was born on 4 Oct. 1993. He received his B.Sc. from Ho Chi Minh Univ. of Sciences in 2015 and his M.Sc. from TUAT in 2019. He is currently a Ph.D. candidate at TUAT. His current research area are handwritten mathematical expression recognition and computer-assisted marking of handwritten mathematical answers.

Cuong-Tuan Nguyen was born on 21 Mar. 1988 in Vietnam. He received his B.Eng. degree from Ho Chi Minh City University of Technology and M.Sc. and Ph.D. degrees from TUAT. Since April 2017, he has been working as an assistant professor at TUAT. He is currently working on handwritten text/math recognition and writer identification.

Masaki Nakagawa was born on 31 Oct. 1954. He received his B.Sc. and M.Sc. degrees from the Univ. of Tokyo in 1977 and 1979, respectively. During the academic year 1977/78, he enrolled in the Computer Science course at Essex Univ. in England and received an M.Sc. with distinction in Computer Studies in July 1979. He received his Ph.D. from the Univ. of Tokyo in Dec. 1988. He has been working at TUAT since April 1979. He is currently a Vice-President at TUAT.