Pattern Recognition - Project 3 on Parsing Handwritten Math Expressions

Sanyukta Sanjay Kate (ssk8153), Pratik Bongale(psb4346)

May 3, 2018

1 Design

We constructed two parsers, the symbol parser and the strokes parser which used the information of strokes from CHROME 2016 dataset. We built a directed KNN graph connecting each symbol to its K nearest neighbors in Euclidean space. Each edge in graph is assigned a weight as maximum probability spatial relationship predicted by parser(multi-class spatial relation classifier). We find a Maximum Directed Spanning Tree(MDST) using Edmond's optimum branching algorithm [1]. MDST algorithm produces a symbol relationship tree(SRT) for label graph file. For stroke parser we built time-series graph with strokes as vertices and use a binary merge/split classifier to predict whether or not to merge connected stroke pairs in graph to form a symbol. We used the same segmenter and classifier from project 2. We identify symbols using this segmenter and pass it to our symbol parser defined above. We used this approach of building a KNN-graph because by finding the K-nearest neighbors we would know the possible number of symbols which a symbol could be connected to. As the KNN-graph would be a directed graph, we implemented the Edmond's algorithm to obtain a maximum spanning tree .

We used geometric features [1], visual features[1], and parzen shape context features[1] for parsing. We had used geometric features because geometric features captures the features related to the different points of the symbols. As we had used geometric features in the previous projects (classification and segmentation) and had obtained high results, we decided to use the same geometric features that we had used previously for parsing. We have used Parzen Shape context features for our parser rather than the multi-shape context features to get a smoother distribution of points in space. By using shape context features, our parser would be trained on the shape of the symbols. We chose not make any changes in our segmenter and classifier that we had previously developed as it had given us good classification and segmentation results respectively. The classifier we used is the random forest classifier and the segmenter we had developed was the time-series graph segmenter [2].

2 Preprocessing and Features

Data preprocessing steps were referred from [3].

a) Duplicate Points Removal Duplicate points were removed from strokes, as repeated points within a stroke do not give any new information about the stroke.

b) Smoothing We commonly observe writing jitter in handwritten data collected with a digital pen, smoothing the stroke points can avoid this type of noise in data. So we smooth every point in every stroke, by taking an average of the current point, the previous point and, the next point in the stroke. The current point is replaced with the new averaged point.

$$avg = \frac{p_{i-1} + p_i + p_{i+1}}{3}$$

where i is a point at time t in each stroke.

c) Normalization We normalized every stroke's y coordinate in the expression between 0 and 100 while maintaining the aspect ratio of the expression. Normalizing every stroke was necessary as every stroke were of different size.

$$y = \frac{y - y_{min}}{y_{max} - y_{min}} * 100$$

$$x = \frac{x - x_{min}}{x_{max} - x_{min}} * 100$$

The x coordinate of the points in the stroke needs to be changed as per the change in y coordinate and hence, the below equation shows how the x coordinate of the points were changed.

d) Resampling The expressions which were provided to us were equidistant in time but not in space. To get the expressions equidistant in space, we had to resample every stroke in the expression to 30 points. We fit a bspline curve of degree 3 over every stroke and sampled 30 points on this curve to get smooth corners in our stroke. Also, bspline curve provides better representation of complex curves in a stroke by adding more points at curvatures as compared to smooth parts of a stroke.

We referred to [1] to build geometric features, visual features and the shape context features used for recognizing the spatial relationships between the symbols. In a symbol pair, the current symbol is considered to be the parent symbol and the other symbol is said to be the child symbol. For every symbol pair, we found six geometric features and one twenty Parzen shape context feature.

Geometric Features: To compute the geometric features, the bounding box centers and the average centers were computed.The corner points of the bounding box for a symbol were computed using the minimum and maximum values of x and y coordinate of that symbol. Using one of the corner points of the bounding box which was (x_{min}, y_{min}) and the width and height of the bounding box, we found the bounding box center using the equation

$$bb_{center} = (x_{min} + w \times 0.5, y_{min} + h \times 0.5)$$

where, w = width and h = height of bounding box.

a) Horizontal Distance between the Bounding Box Centers b) Vertical Distance between the Bounding Box Centers c) Distance between the Bounding Box Centers d) The distance between the averaged centers of the symbol pairs e) Writing Slope

$$cos\theta = \frac{ab^2 + bc^2 - ac^2}{2 \times ab \times bc}$$

Here, a is the last point of the parent symbol, b is the first point of the last stroke of the child symbol, c is the point on the horizontal line. ab, bc and ac represent the distances between the points a, b and c. We then took the inverse of the value acos(cos θ) we got from the above equation to get the slope.

f) The maximal point pair distance

Parzen Shape Context Features: We used parzen window shape context features as described in [1]. We decided to use parzen shape context features instead of symbol pair multi-shape context feature which we used in project 2. This is because in parzen shape context feature, every symbol point contributes to every bin in the histogram, thus giving a smoother distribution of the symbol points over all bins. Two circular histograms were built, one around bounding box center of a parent symbol and the other around bounding box center of child symbol. The circular histogram was divided into sixty bins, hence, the total number of parzen shape context features were one twenty features. In the multi-shape context features which we had used for our time-series segmenter, there were too many empty bins which did not contribute much in describing the shape of the strokes. This was not the case with parzen window shape context features. In parzen window features a 2D gaussian distribution was used as a kernel function for each bin, which used all the points present in the histogram in order to get a continuous distribution of the points. We referred to the equation of 2D probability distribution function from [1]. If the bins were far away from a point, then the probability density of that bin was very low than the bins which were near to the symbol points.

3 Parsers

Symbol Parser:

Read symbols from inkml file and ground truth relationships from lg files with objects and relationships (six relations) between them. Each symbol pair can have one of the six possible relations. We have not used any grammars for our parsers as we have created a graph based parser.

For the symbol parser, we built a K-nearest neighbor graph in which the symbols were the vertices and each vertex was connected to K-nearest symbols with an edge. The distance between the symbols were calculated using the Euclidean distance. We chose K = 2 initially, and then after observing the KNN-graph results, we realized that there were a lot of chances that a symbol may not consider all its nearby neighbors. Even with K = 4, a symbol may not choose all it's nearby symbols and hence, we thought having K = 6 would cover maximum nearest neighbors for a symbol. Therefore, we chose K = 6. Here, a symbol pair would be two vertices linked to each other via an edge. Each edge is then assigned a maximum probability and a relation predicted by the parser

(multi-class spacial relationship classifier). We give the KNN-Graph as an input to the Edmond's algorithm in order to get the Maximum Spanning Directed Tree which is the Symbol Relationship Tree (SRT) for the label graph file. The algorithm which we have implemented for the Edmond's algorithm requires the input graph to be a strongly connected graph and hence, if the KNN-Graph obtained is not a strongly connected graph then we make it strongly connected by adding in extra edges (connecting every vertex to the other vertices in the KNN-graph). The output of the Edmond's algorithm will be a Symbol Relationship Tree.

Pseudo code for the Symbol Parser

```
Inputs:
S: the set of segmented symbols forming an expression
Parser: trained parser model (multi-class classifier with 6 classes)

if only one symbol:
    write sym as it is in the lg file

for each symbol in S:
        let knn_graph be the KNN with K=6 nearest symbols(by Bounding Box Center)

         if knn_graph not strongly connected graph:
              make knn_graph fully connected

         for vertex u in knn_graph:
               for every neighbor v of vertex u:
                   let e be an edge between u and v
                   let feat be the feature vector with geometeric and parzen context features

                   let pr be the probabilities predicted by spatial relationship parser model
                   let cl be the class with the highest probability

                   let max_prob be the maximum probability from the pr vector
                   assign edge e weight of max_prob along with it's relationship class cl

      let edm_graph be the graph which stores all the max_prob on the edges
      let root vertex be the leftmost symbol in expression
      get a maximum directed spanning tree using edmonds algorithm

      let mst_srt be the symbol relationship tree obtained from Edmond's Algorithm
      write mst_srt into an lg file which will have objects and relationships
```

Time complexity analysis of the symbol parser:

- The time complexity of constructing KNN Graph is $O(N^2)$ where N is the number of symbols in expression.
- The time complexity for checking if the KNN graph is strongly connected is O(V+E) as we have used Depth first search, where V is the number of symbols and E is the number of edges in graph.
- To get the features for symbol pairs (current symbol and it's neighbor), the time complexity is O(NM) where N is the number of symbols and M is the maximum number of neighbors a symbol has.
- The time complexity of the Edmond's algorithm is O(ElogV) as described in the fast implementation we referred from article [1]

Hence, the time complexity of the symbol parser for one expression is $O(N^2)$.

For the stroke parser, we took the strokes from the inkml files as input. We used the non-baseline segmenter (time-series graph) from Project 2 for generating the symbols/segments. Using the segments built by our segmenter, we use our symbol parser to classify the spatial relationships between the segments. We write all the objects and their relationships into an lg file. The symbol passing after we have obtained the segments is same as the approach described above in the symbol parser.

Pseudocode for the stroke parser

```
Inputs:
```

```
S, the set of handwritten strokes for an expression
Parser, trained parser model (multi-relationship classifier with 6 classes)
SEG, trained segmenter model for the segmentation
CLF, trained classifier model for classification

for strokes S in an expression :
   let segs be the segments obtained from the time-series segmenter
   CLF is used for classifying the segmented symbols

if only one symbol:
    write sym as it is in the lg file

for each symbol in S:
        let knn_graph be the KNN with K=6 nearest symbols(by Bounding Box Center)

         if knn_graph not strongly connected graph:
              make knn_graph fully connected

         for vertex u in knn_graph:
               for every neighbor v of vertex u:
                   let e be an edge between u and v
                   let feat be the feature vector with geometeric and parzen context featu

                   let pr be the probabilities predicted by spatial relationship parser mo
                   let cl be the class with the highest probability

                   let max_prob be the maximum probability from the pr vector
                   assign edge e weight of max_prob along with it's relationship class cl

Same as the symbol parser from here
```

The complexity of the stroke parser is same as the symbol parser. Here, the time complexity of considering segmenting the strokes into symbols will also be considered. The time complexity of segmenting the strokes into symbols is $O(S^2)$, where S is the number of strokes in an expression. As the time complexity defined for symbol parser is $O(N^2)$, the overall time complexity of the stroke parser for one expression will be $O(S^2)$, as parsing the stroke pairs will be more than parsing each symbol twice. If the number of strokes will be equal to number of symbols, then in that case the over all time complexity will be $O(N^2)$, where N is the number of symbols in the expression.

4 Results and Experiments

Data Split: We extracted individual symbols from given dataset of inkml files and computed the priors for each symbol in ω = set of isolated symbol classes. These priors were split into two parts 2/3rd for training and 1/3rd for testing thus obtaining two probability distributions $P_{tst}(x)$ and $P_{trn}(x)$ where $x \in \omega$. We need to split the collection of inkml files such that we maintain the prior distribution as close to the above distributions $P_{tst}(x)$ and $P_{trn}(x)$ as possible. The closeness is computed using KL divergence.

$$DKL(P||Q) = \sum_{i=1}^{N} p(xi) \ . \ (log\ p(xi)\ -\ log\ q(xi))$$

The value of DKL(P||Q) provides as estimate of information lost if we us an approximation(Q) instead of original distribution(P), so a lower value of DKL is favorable. We make 1000 random splits and find best value of divergence. We obtained best DKL = 0.0003229.

We created a symbol parser and a stroke parser and recorded its results. We tested our parsers on the ground truth test dataset. For symbol parser we observed that the symbol segmentation and classification was 100% as they were directly obtained from the inkml files. For the stroke parser we observe that the f-measure for segmentation was 88.29 % and with classification it was 59.05 %. The symbol parser gave an f-measure of 35.37 % for classifying the spatial relations. The stroke parser gave an f-measure of 28.42 % for classifying the spatial relations between the symbols obtained from the time-graph segmenter.

	$F-mesaure(\%)$
Stroke Parser (Relations) on test set	28.42
Stroke Parser (Relations+Classes) on test set	27.85
Symbol Parser (Relations) on test set	35.37
Symbol Parser (Relations+Classes) on test set	34.66
Symbol Parser with Resub on training set(Relations+Classes)	38.38

Table 1: Excerpt from Summary.txt

We observe from the Table 1, that the F-measure for the stroke pair along with classification is even lesser than the symbol parser for the test dataset. This is because we have used our own segmenter and classifier to classify symbols and then run our parser on these segmented symbols. Whereas, in the symbol parser, the segmentation and classification of the symbols is 100%, thus increasing the f-measure for the recognizing the spatial relations between the symbols.

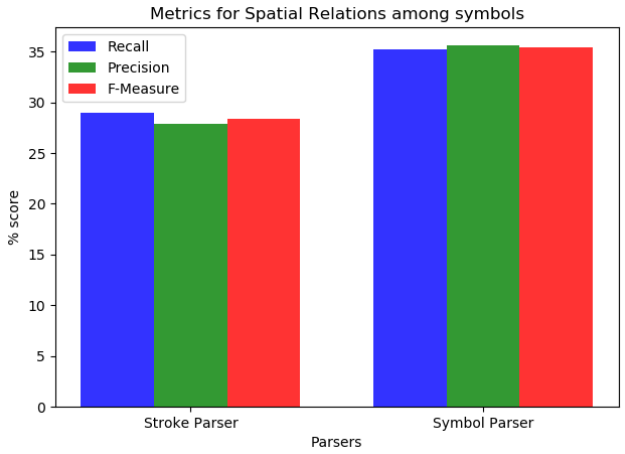

Figure 1: Performance metrics for recognizing spatial relations for the parsers

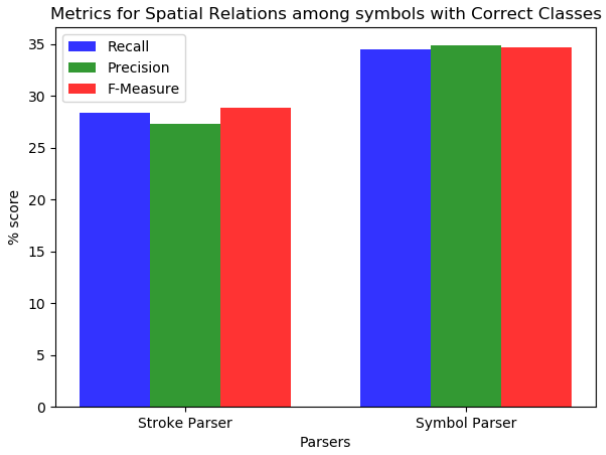

Figure 2: Performance metrics for recognizing spatial relationships with classes for the parsers

From the above figures we observe that the recall and precision for the testing dataset was 35.17% and 35.17% respectively for symbol parser. Whereas, for the recognizing the spatial relationships, the precision and recall for the stroke parser was 28.94% and 27.92% respectively, which is lesser than the symbol parser.

Below Figures are the results obtained from the lg2dot. The expression is { T }

We see from figure 3, that our stroke parser was able to segment and classify the symbols correctly, but has

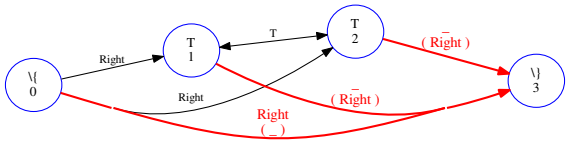

Figure 3: Expression which is recognized using the stroke parser

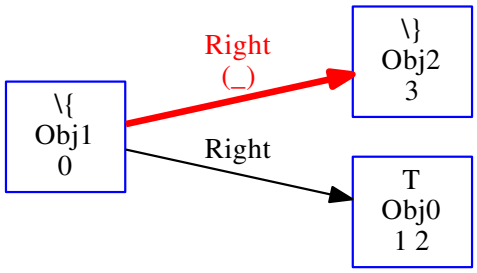

Figure 4: Expression which is recognized using the symbol parser

failed to recognize the correct relation between 'T' and '}'. This is because the probability obtained for the edge between 'T' and '}' is lesser than the probability obtained for '{' and '{'. Therefore, the Edmond's algorithm chose the edge with the higher probability weight. It is true that '}' is to the right of '{', but this is not what is present in the ground truth of the expression and hence, the relation between '{' and '}' is considered incorrect. Our K-NN algorithm selects the 6 nearest neighbor, in this case just two nearest neighbors for each symbol and hence, the edge between '{' and '}' is also taken into consideration. Our K-NN graph algorithm considers many such edges which are unnecessary. To avoid unnecessary edges, or to improve our parser we can implement the 'Line of Sight Graph' [1] which only considered the edges between the symbols which are visible to each other. This will help not taking into account the unnecessary edges.
We see in the symbol parser, the same issue occurred which had occurred for the stroke parser.

When the results of Confusion histogram for the strokes parser (with symbol pairs) were observed, we observed that the segmentation, that is merging and splitting of the symbols in a lot of expressions have taken place correctly, but the classification of those merged strokes (symbols) have failed at times. We also see errors in the relationships between the segmented symbols. For example, consider an expression which has a relation of 'Sup' from symbol 'x' (made out of two strokes) and symbol '2'. We observe 20 errors where the parser has failed to recognize any relation between 'x' and '2' but has managed to merge the two strokes of symbol 'x', thus segmenting the symbol correctly. Further, the symbol '2' was also misclassified as symbol 'y' rather than symbol '2'. We observed that there were around 7 errors in which the symbol '2' was misclassified as symbol 'n', but there no errors in segmentation and recognizing the relationships ('x' was linked to '2' as 'Sup'). We found this example interesting, as it showed us that even though there might be classification errors, the classification of relations and segmentation is correct.

5 References

1. R Zannibi and L Hu, "Segmenting Handwritten Math Symbols Using AdaBoost and Multi-scale Shape Context Features", Document Analysis and Recognition (ICDAR), 2013 12th International Conference, pp: 1180-1184.
2. R Zanibbi and L Hu, "Segmenting Handwritten Math Symbols Using AdaBoost and Multi-Scale Shape Context Features", Document Analysis and Recognition (ICDAR), 2013 12th International Conference.
3. R Zanibbi and L Hu, "HMM-Based Recognition of Online Handwritten Mathematical Symbols Using Seg-

mental K-Means Initialization and a Modified Pen-Up/Down Feature", Document Analysis and Recognition (ICDAR), 2011 International Conference.