Project 2: Segmentation

CSCI 737: Pattern Recognition

Anushree Das (ad1707),
Nishi Parameshwara (np9447),
Omkar Sarde (os4802)

April 21 2021

1 Design

Our classification and segmentation model is designed as given below:

We have parsed the .inkml files using xml Element tree to get stroke trace coordinates, stroke id, stroke symbol, stroke expression, and UI. We performed pre-processing on the stroke trace coordinates obtained after parsing the files. The data pre-processing steps are similar to [1]. We completed only the necessary three pre-processing steps from [1]: removing duplicates of stroke points, size normalization of the stroke points, and resampling the stroke points. We removed duplicate points as we realized it added to the redundancy. After observing the trace coordinates in space, we found out that they were not equidistant from each other and had inconsistent sizes. So we performed normalization to achieve equidistant points. And finally, to ensure that enough points represent each stroke, we applied the pre-processing step of sampling to get 30 points for each stroke to get a consistent representation.

For the segmentation model, we referred [2]. We have implemented Line of Sight graphs (LOS), amongst the various graph representations, to understand and interpret the math formula better. As part of the segmentation technique, we needed to decide whether to merge some strokes to form a symbol. To aid this decision, we also extracted geometric features and implemented Parzen Shape Context (PSC) to get the features and shape of the strokes that could represent the symbols better, making it a crucial feature extraction step for this model. For classification, we have decided to go ahead with classifiers implemented in Project 1. Specifically, we are using the Random Forest classifier and Support Vector Machines for classification as it gave exceptional accuracies for our Project 1.

2 Pre-processing and Features

In this classification model, we followed pre-processing steps described in [1]. They are listed below:

- Duplicate point filtering
  Removed redundant and repeated points in the stroke data points. We need to remove it to generalize the model and not confuse it as the duplicate points do not convey any important information.

- Size normalization
  Scaled coordinate values between 0 and 1 for y-axis and modified the x-axis with respect to the aspect ratio of the image file. The formula used were:

$$y = \frac{y - y_{min}}{y_{max} - y_{min}} \tag{1}$$

$$x = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2}$$

  Due to the formula used for normalization and our implementation of it, we come across mathematical errors, especially when there are not many stroke coordinates or repeated values of stroke coordinates. It led to division by 0 error. For this reason, we modify the formula to consider this edge case when the length of the points is less, or we get the same values. This modification is achieved by initially incrementing the maximum y coordinate value with a positive number and decrementing maximum x values with the same positive number.

- Resampling
  To ensure that all the points are equidistant from each other in time and space along the original trajectory of the symbol, we need to resample each symbol to have 30 stroke points. So, we must calculate that for each symbol to 30 points, each stroke must have a fixed set of points and adjust accordingly. Then we try to fit the points to get the perfect curve after resampling. While fitting the curve, we came across a problem where the resampling function gave an error while fitting the curve when the stroke points are less than 4. So, to solve this issue, we added more points.

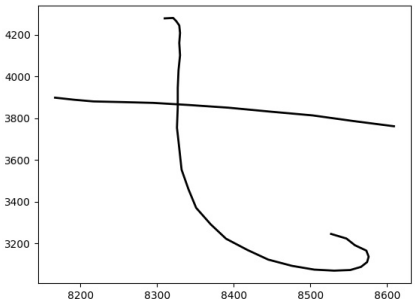

Figure 1: Sample Symbol before pre-processing

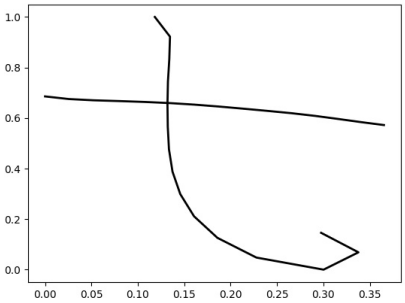

Figure 2: Sample Symbol after pre-processing

3 Symbol Classifier

After pre-processing, we use the updated stroke information to extract features. These features are the same as mentioned in [1] i.e., normalized y-coordinate, vicinity of the slope, and curvature. We calculate three features for each point of every symbol, and we have 30 points per symbol, so 90 features are extracted for each symbol in one data file.

1. Normalized y-coordinate
   Size normalization obtained during pre-processing is used as one of the features.

2. Vicinity of slope
   Tangent between the straight line joining points $x(t-2), y(t-2)$ and $x(t+2), y(t+2)$ and horizontally across point $x(t-2), y(t-2)$. Consider the figure 3 from [1] to understand this angle. Here, tangent of alpha is the vicinity slope.

3. Curvature
   Tangent between the straight line joining points $x(t2), y(t2)$ and $x(t), y(t)$ and the straight line joining point $x(t), y(t)$ and point $(x(t+2), y(t+2))$.

In this Project we are using the same Random Forest Classifier from project 1.

3.1 Random Forest Classifier

A random forest fits several decision tree classifiers by taking samples of the data set. It uses averaging to increase accuracy and reduce overfitting. It is made up of a large number of decision trees that work as an ensemble. Each tree in the random forest gives its class for prediction, and the majority vote decides the winner. For our model, random forest classifier taken from ensemble section of scikit- library takes the following parameters: min_samples_split = 3, n_estimators = 50, max_depth_size = 40, max_features = 11.

4 Segmentation

Here we have described the 16 geometric features and shape context features that we have used for the implementation of our segmenter.

4.1 Geometric Features

4.1.1 Horizontal distance between stroke pairs

This the average distance between the x-coordinates of the stroke pairs.

4.1.2 Size difference between stroke pairs

This is the absolute difference of bounding box area of stroke pairs.

4.1.3 Vertical Offset

This the average distance between the x-coordinates of the stroke pairs.

4.1.4 Minimum Point Distance

This is the minimum distance between two points.

4.1.5 Overlapping area of bounding boxes

This gives the overlapping area of bounding boxes of stroke pairs.

4.1.6 Minimum distance between boxes

This is the minimum distance between two bounding boxes.

4.1.7 Horizontal Overlapping of bounding boxes

This gives the absolute horizontal overlapping distance of bounding boxes.

4.1.8 Distance and offset between stroke start and end points

These are four separate features i.e. distance for starting point of stroke, distance for ending point of stroke, offset for starting point of stroke and offset for ending points of stroke.

4.1.9 Parallelity

This is the angle between two vectors representing strokes.

4.1.10 Distance between bounding box centers

This is the distance between centres of bounding boxes.

4.1.11 Distance between centers-of-mass

This is the distance between centres of mass of bounding boxes.

4.1.12 Maximal Point pair distance

This is the maximum point of distance between two strokes i.e. maximum distance between the point on the current stroke and point in the next stroke.

4.1.13 Horizontal offset end point and start point

This is the horizontal offset between last point of the stroke and first point of the starting stroke.

4.1.14 Vertical distance between bounding box centers

This is the vertical distance between the bounding box centers of stroke pairs.

4.1.15 Writing slope

This is the angle computed by the line connecting the first and the last point and the horizontal line emerging from the last point of the current stroke.

4.1.16 Writing curvature

The angle between the lines defined by the first and last points of each stroke.

4.2 Parzen Shape Context Features

From [2], Shape Context is the relative position and pixels in an image around a given point using a log-polar histogram. We use Shape Contexts to characterize the pixels in an expression image around two strokes being considered

for merging. First, with Parzen window estimation using a 2D gaussian kernel, we produce smoother probability distribution. Second, the shape context region is divided into bins using uniform angles and distances from the center of the histogram. The center of the PSC is the average of the two stroke bounding box centers. The radius of the shape context includes the strokes being compared.

We take pairs of strokes and find geometric features and the shape context features for this stroke pair. So for x strokes, we have x-1 segments initially. The decision to merge a stroke pair is provided by a our binary classifier (segmenter). We merge stroke pair who have a common stroke and also consider remaining individual strokes as one segment. These identified segments are passed to a classifier to predict their labels and write the prediction output for each segment to label graph file.

5 Results and Experiments

Segmentation and Classification Algorithm Results

Type	Recall	Precision	F-measure
Objects	67.64	51.36	58.38
+Classes	39.06	29.66	33.72

S-Oracle Results

Type	Recall	Precision	F-measure
Objects	65.82	46.60	54.57
+Classes	65.82	46.60	54.57

AC-Oracle Results

Type	Recall	Precision	F-measure
Objects	91.44	84.75	87.97
+Classes	91.44	84.75	87.97

Top Three Errors for Segmentation and Classification Algorithm

No.	Symbols	Errors
1	+	4947
2	x	4716
3	=	3483
Total		13146

Top Three Errors for S-Oracle

No.	Symbols	Errors
1	+	5381
2	x	4545
3	=	3588
Total		13514

Top Three Errors for AC-Oracle

No.	Symbols	Errors
1	sin	550
2	i	490
3	x	453
Total		1493

5.1 Analysis of Results

The segmentation and classification algorithm utilized by us achieved F-measure scores of 58.38% and 33.72% for segmentation and for segmentation and classification (combined) respectively. It ranked second for segmentation, while it came last for segmentation and classification combined. This can be attributed to the fact that the random forest classifier is misclassifying more symbols when coupled with the segementer from the paper [2]. From assignment 3 results for k-rec and ac-rec respectively, we can infer and attribute this drop in accuracy to the random forest classifier. This might be due to the fact that necessary input features needed for accurate classification were not available from the clusters which the segementer output. The S-Oracle achieved F-measure score of 54.57% for segmentation and for segmentation and classification (combined), whereas AC-Oracle had the highest F-Measure of 87.97% for both the metrics. From the segementation results we can ascertain that our algorithm formed better clusters than the hypersegmented clusters of S-oracle but failed in comparision to the results of agglomerative clustering.

Analysis of ConfHist results mimic the F-measure scores, where in our algorithm is comparable to the S-Oracle in terms of total errors and has the same symbols as the top three errors. The top three errors contain multi-stroke symbols like '+', 'x' and '+'. Further analysis reveals that these

symbols have the highest false association rates, as in for a cluster for a certain symbol, an additional stroke was added making the classifier misclassify the symbol. AC-oracle, due to its better segmentation, had 1/10 of the total errors but depicted the same false association trends.

References

[1] L. Hu and R. Zanibbi. 2011. HMM-Based Recognition of Online Handwritten Mathematical Symbols Using Segmental K-Means Initialization and a Modified Pen-Up/Down Feature. In *2011 International Conference on Document Analysis and Recognition*. 457–462. https://doi.org/10.1109/ICDAR.2011.98

[2] Lei Hu and Richard Zanibbi. 2016. Line-of-Sight Stroke Graphs and Parzen Shape Context Features for Handwritten Math Formula Representation and Symbol Segmentation. In *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 180–186. https://doi.org/10.1109/ICFHR.2016.0044