Pattern Recognition - Project 2 on Segmentation

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

April 12, 2018

1 Design

The input data is in the form of x, y point co-ordinates for each stroke of online handwritten mathematical expressions written by different authors or generated by a grammar [3]. On observing the input inkml files, we found that strokes included several duplicate points which was just redundant data. After removing these duplicate points, we visualized expressions by plotting stroke points and learned that these points are not equidistant in space and had varying width and height, so we performed smoothing and normalization as described later. Also, in some cases, strokes were represented with too many points while others had too few points. To have a consistent representation, we resampled every stroke to have 30 points.

The decision problem of whether to merge a set of strokes to form one object requires features describing the geometry and shape of strokes in question. So we chose to use geometric features to represent the shape of segments along with context features to obtain better sense of the context of symbols. To restrict the number of segmentation candidates generated, we only considered successive strokes in time. Given N strokes we computed features for N-1 stroke pair and trained our segmenter to determine which stroke pairs to merge and which to split. We chose this approach for its simplicity and time efficiency.

The symbol classifier we used is the this project is the same as the one we used in project 1. We decided to use the same model as it obtains expected results in terms of individual symbol classification(as per our evaluation results).

2 Preprocessing

We refered [2] for deciding on data preprocessing.

a) Duplicate Points Removal Duplicate points were removed from the 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.

3 Symbol Classifier

We have used Random Forest classifier and did make any changes with respect to features or parameters of random forest(number of estimators=100, max depth=15). We retrained this classifier on the symbols in training split of the new dataset provided for project 2. The features used by this classifier are cosine of slope, sin of curvature and normalized y-coordinate as described in [HMM-Based Recognition of Online Handwritten Mathematical Symbols Using Segmental K-Means Initialization and a Modified Pen-Up/Down Feature]

4 Segmentation

4.1 Features

This section of the report describes the geometric features and the shape context features for the segmenter we have implemented. Each stroke pair (the current stroke and it's successive stroke in time series) gives six geometric features and sixty features describing the shape's context as described in [1].

4.1.1 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 stroke were computed using the minimum and maximum values of x and y coordinate of that stroke. 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 This feature is calculated by taking an absolute value of the difference between the x coordinates of the bounding box centers of the stroke pairs.

b) Vertical Distance between the Bounding Box Centers This feature is calculated by taking the absolute value of the difference between the y coordinates of the bounding box centers of the stroke pairs.

c) Distance between the Bounding Box Centers We used euclidean distance to calculate the distance between the bounding box centers of the stroke pairs.

d) The distance between the averaged centers of the stroke pairs the average center of the stroke pairs were calculated by taking the average of the x-coordinates and the y-coordinates of the strokes. Euclidean distance was used to compute the distance between the averaged centers of the stroke pairs.

e) Writing Slope The last point of the current stroke, the first point of the next stroke and a point on the horizontal line emerging from the last point of the current stroke. We compute angle formed by the line connecting the first and the last point and the horizontal line. This angle is known as the writing slope. To compute this angle, the cosine of the angle was found using the equation

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

Here, a is the last point of the current stroke, b is the first point of the last stroke, 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 The maximal distance between two strokes is the maximum distance between one point on the current stroke and the other on the next stroke. We took every point from the current stroke and found the euclidean distance between that point and every point on the next stroke. The maximum distance was recorded and considered as a feature for our segmenter.

4.1.2 Shape-Context Feature

We implemented the stroke-pair shape context feature to capture the shape of the stroke pair. The circle of a certain radius is drawn with the center as the center of the bounding box of the current stroke. The radius is the maximum distance between the center of the bounding box and one of the points on either the current stroke or the next stroke. The circle formed from this radius contains either all points from both the strokes or just few. The circle is divided into 12 parts with 4 concentric circles (excluding the outer circle), thus breaking the circle into 60 parts. Each part is known as a bin. The number of points in each bin is calculated and divided by the number of points in the entire circle and is termed as the normalized count. This normalized count of the 60 bins was considered as 60 features for that stroke pair.

4.2 Segmentation Algorithm

We have used the technique used by Hu and Zannibi in [1].

Pseudo code

```
Inputs:
S, the set of handwritten strokes for an expression
SEG, trained segmenter model (binary classifier: 1-Object, 0-Not Object)
CLF, trained classifier model for symbol classification

for each consecutive stroke pair ($s_i$,$s_{i+1}$) $\in$ S, do
    let gf be geometric features extracted from stroke pair (s1,s2)
    let sf be shape context features extracted from stroke pair (s1,s2)
    let pred be prediction by SEG for feature vector <gf,sf>

    if pred is 1:
        add (s1,s2) to merged strokes set M

    for each set $m_i$ $\in$ M, do
        while $m_i \cap m_{i+1}$ is not $\emptyset$
                merge sets $m_i$ and $m_{i+1}$

    let SS be a set of predicted symbols
    let SS = SS $\cup$ M
    for s $\in$ S, do
        if s $\notin$ SS
                let SS = SS $\cup$ M

    let G be an empty label graph
    for y $\in$ SS, do
        let l be CLF prediction of label for symbol y
        add object y and label l to label graph G

return G
```

Description We take two successive strokes in time and find geometric features and the shape context features for this stroke pair. So for N strokes we have N-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(finding connected components) 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.

The above segmentation strategy was chosen based on the simplicity and time efficiency it offers.

We used a random forest classifier as our detector to determine whether given strokes make an object(segment). This random forest model uses 100 estimators with maximum tree depth of 20. These parameters gave the best results in our grid search for parameters. We did not observe significant difference on increasing the number of estimators in our random forest model. The symbol classifier used is also a random forest model with 100 estimators and maximum tree depth of 15.

5 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 baseline segmenter to compare with our more sophisticated segmenter. In baseline segmenter we just considered one stroke at a time with three features per point on every stroke and fed it to our classifier to predict the symbol. Our sophisticated segmenter made use of two successive strokes in time series, merged the necessary strokes and then were fed into the classifier. We observe that the sophisticated segmenter works better than the baseline classifier. This is because, no combination of strokes is considered in the case of baseline segmenter and hence, the strokes are not merged together to form a symbol. Therefore, there is a lot of over-segmentation as even a symbol with two or more strokes is considered to be more than one symbol. When the lg files which are created for the baseline classifier are compared with the ground truth files, the F-measure or the accuracy comes to 54.64% for the test dataset for the objects or symbols. The classification results for the baseline segmenter is 36.42%. The results for the classification is less again, because for classification, each stroke is considered to be a symbol, and when each stroke's class is predicted, it gives a wrong predicted value as the classifier is trained on the provided symbols from the inkml files.

The metrics results for the sophisticated segmenter is given below.

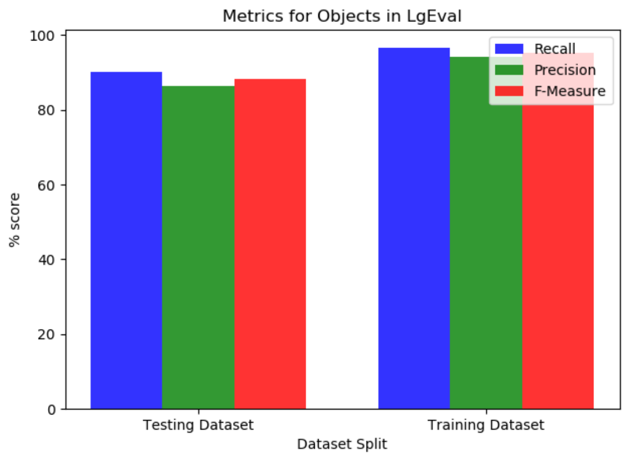

Figure 1: Performance metrics for symbols in label graphs

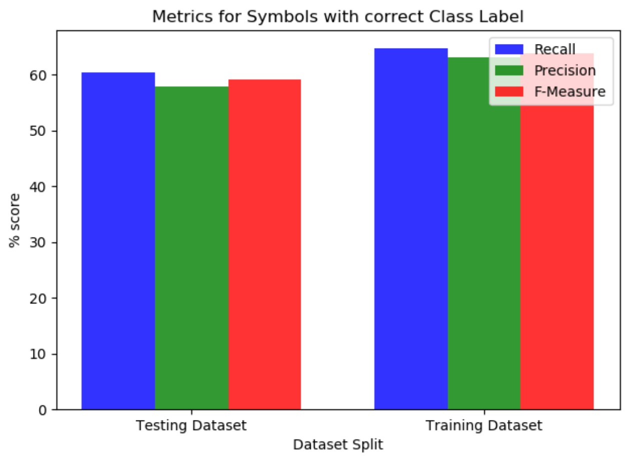

Figure 2: Performance metrics for symbols with correct class labels

We see the F-score for the sophisticated segmenter obtained for the testing dataset is 88.3% and for the training dataset it is 95.26% for the Objects in the LgEval summary file. The precision and recall for the testing

Predicted \ Ground Truth	=	)	z
+	92	4	5
−	93	0	3
n	0	4	243
y	1	25	488
1	4	349	6

Table 1: Excerpt from confusion histogram

dataset was 86.44% and 90.24% respectively. Whereas, for the symbols with the correct class label was 59.17% which is less than the accuracy obtained using just the segmenter.

In Node Label Confusion matrix for the test dataset, we see that the number of errors are 17310. The below table is an example describing the confusion matrix of the errors of the segmenter.

We observe from the above table that there are symbols which are classified wrongly a multiple times with different symbols. For example, ')' is classified as '1' 349 times. We see this error because we need to train our segmenter on more context shape features. As '(' looks a bit similar to '1', the segmenter needs more context shape features such that it can capture the neighborhood and global context shape features.The context shape features used by our segmenter only looked at the stroke pairs rather than considering it's neighboring strokes and the entire expression. Hence, we get a lot of errors while segmenting '(' as '1'. We see that '=' is classified as '-' 93 times.
In the Confusion Histogram we observed the number of errors which had occurred while segmenting different symbols drawn with different number of strokes. For example, the symbol 'x' which were created using different number of strokes, were segmented as different objects a multiple times. For example, when x draw with two strokes, were segmented as 'y' 98 times, '×' 95 times, and even the two strokes considered for x were segmented as two different symbols, such '(' and 'y' 5 times. These errors occurred because 'x' is similar to '×' in shape, and hence, we need more features describing the shapes and geometry of the strokes of 'x'. When a single stroke is overlaps with any of the other strokes then, segmenting that one stroke as a symbol would be difficult and would lead to under-segmentation, thus leading to predict a different symbol such as '6' symbol was classified as '0'. We do not see a lot of over-segmentation errors, for example, in few symbols such as $'\cos'$ (written with 3 strokes) were over-segmented as 'a' and 'b' 8 times. We see a large amount of errors in symbols which are a bit similar to other symbols, such '(', ')', '1', '6', '0'.
We observed from our confusion histograms, that symbols such as '≤', 'cos', 'Y', 'P', 'E', '≥', '∃' gave less errors, thus improving the recall and precision of our system. There were few symbols such as 'Y' and 'P' which were segmented and classified as 'y' and 'p' respectively. This shows that the segmenter got confused between the upper case letters and the lower case letters. To avoid this confusion, the segmenter must be trained on different more symbols of upper case and lower case letters with more shape features. We have not made any use of prerecognition in our system to avoid the over-segmentation of the overlapping symbols and hence, we plan on improving our system (recall and precision) with prerecognition for overlapping symbols [1].

6 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.

[2] R Zanibbi and L Hu, "HMM-Based Recognition of Online Handwritten Mathematical Symbols Using Segmental K-Means Initialization and a Modified Pen-Up/Down Feature", Document Analysis and Recognition (ICDAR), 2011 International Conference

[3] Richard ZanibbiUtpal GarainChristian Viard-Gaudin, "Advancing the state of the art for handwritten math recognition: the CROHME competitions", International Journal on Document Analysis and Recognition (IJDAR), 2011