Segmenting Line Graphs Into Trends

Peng Wu[1], Sandra Carberry[1], Stephanie Elzer[2]

[1]University of Delaware, Computer and Information Science Department
[2]Millersville University, Computer Science Department
{pwu,carberry}@cis.udel.edu elzer@cs.millersville.edu

Abstract— *Information graphics (line graphs, bar charts, etc.) often appear in popular media such as newspapers and magazines. Such graphics generally have a message that they are intended to convey. Our overall project goal is to extract this message. For a line graph, the first step is to segment the graph into a series of visually distinguishable trends. This paper presents our methodology for identifying this segmentation. We use a support vector machine to produce a learned model of when to split a segment of the graph into subsegments; the support vector machine considers a variety of features, including statistical tests, other characteristics of the segment under consideration, and global features of the graphic. The paper presents three evaluations of our graph segmentation model, which show the effectiveness of our system.*

***Index Terms*— line graph, graph segmentation, trends**

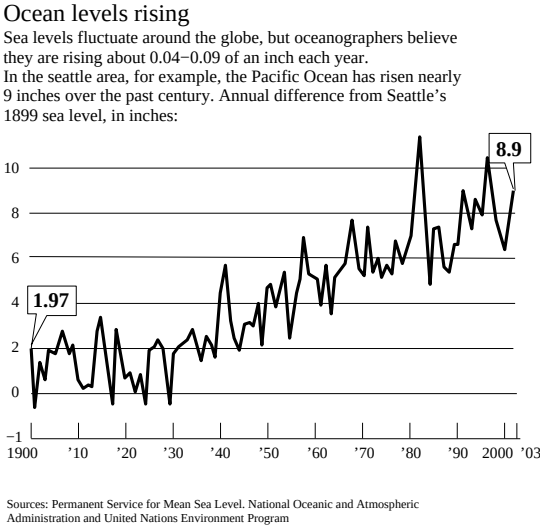


Fig. 1: A line graph from USA Today

1. Introduction

Information graphics (line graphs, bar charts, etc.) are widely used in popular media such as newspapers and magazines. Such graphics generally have a message that the graph designers intended to convey and which captures the high-level knowledge contained in the graphic. For example, the line graph in Figure 1 ostensibly is intended to convey a changing trend in ocean levels from relatively stable between 1900 and 1930 to rising thereafter. Our goal is the reverse of graphic design. We want to develop a system that uses the communicative signals in a graphic (such as coloring or annotations) to recognize the graphic's message and thereby extract its high-level content.

In order to recognize the high-level message conveyed by a line graph, we must treat it as a sequence of visually distinguishable trends rather than as a large set of data points connected by short line segments. For example, Figure 1 shows two visually distinguishable trends for ocean levels — a relatively stable trend from 1900 to 1930 and a rising trend from 1930 to 2003 (both with high variance).

This paper presents our model for segmenting a line graph into a set of visually apparent trends. The model is constructed by a support vector machine that takes into account both local and global attributes. The advantage of using machine learning to produce the graph segmentation model is that the machine learning algorithm can consider a variety of candidate attributes and emphasize those that are important in producing a segmentation that captures trends which are visually apparent to humans, as opposed to using an algorithm that comes from the perspective of error minimization. Our model of graph segmentation will be used in a Bayesian network that reasons about the communicative signals in the graphic (such as a point being annotated with its value) to hypothesize the graphic's high-level message.

To our knowledge, our work is the first approach to graph segmentation that 1) captures trends that are visually apparent to humans, 2) uses both global and local information, and 3) uses machine learning to produce a learned graph segmentation model.

Section 2 describes our approach to building a model of graph segmentation. Section 3 presents several examples of segmentations produced by our system, and Section 4 presents our evaluation experiments, including a cross-validation of our decision module that determines whether to split a segment, a comparative evaluation with another method, and an evaluation of the quality of our segmentations. Section 5 discusses related work and Section 6 is our conclusion.

This material is based upon work supported by the National Science Foundation under Grant No. IIS-0534948 and by the National Institute on Disability and Rehabilitation Research under Grant No. H133G080047.

2. Problem Formulation and Algorithms

2.1 General framework

Given a series of sampled data points from a line graph, we need to segment this data set into one or several sequences where each sequence of sample points can be represented by either piecewise linear interpolation or piecewise linear approximation[1] to show a visually distinguishable trend.

The future application of our research requires that we use a method which can accomplish three goals:

1) A fast algorithm so that it can be used in a real-time system to do the graph segmentation.
2) Some line graphs can be very smooth, but others can be very jagged with large variance, as in Figure 1. Rather than having a set number of segments or threshold of error, our algorithm must deal with a wide variety of line graphs and determine the number of segments as it produces the segmentation for a particular graphic.
3) The existing algorithms for time series segmentation are mainly based on error reduction methods. Since our goal is to identify a segmentation that captures human perception of visually apparent trends, we must use machine learning to consider a variety of different attributes and produce a learned model that emphasizes the most important attributes in identifying visually distinguishable trends.

We have chosen a top-down approach for our segmentation task[1]. An advantage of the top-down approach for our purpose is, as the segmentation moves from the whole graph to individual segments, it is possible to record global information about the larger graphic and pass it for consideration when analyzing the subsegments and deciding whether to divide them further.

Our segmentation algorithm is a recursive algorithm that starts from the whole line graph as one segment. Given a segment as input, the decision module makes a split/no-split decision. If a split decision is made, the splitting module will determine the splitting point, split this segment into two subsegments, and call the decision module on each new subsegment. Recursion stops when the decision module makes a no-split decision on a segment.

The splitting module and decision module are covered in Section 2.2 and Section 2.3 respectively.

2.2 Splitting Module

The splitting module is responsible for selecting the splitting point for each segment once the decision module determines that the segment should be split. Fu-lai Chung et. al.[2] introduced a simple method which uses the PIP (perceptual important point) as the split point in a segment. The idea can be simply described as finding the point which has the largest perpendicular distance from the straight line which connects the two endpoints of the segment.

Choosing the PIP as the splitting point is only complexity $O(n)$, as opposed to choosing a split point that minimizes the sum of squared errors which would be $O(n^2)$. However, human perception is sometimes more sensitive to the maximum or minimum point than to the PIP. If two points are close to one another, with one being the PIP P_{PIP} and the other being the maximum/minimum point P_M, people usually choose the maximum/minimum point as the end of one trend segment and the start of the next trend segment. Thus we consider both the PIP and the maximum/minimum points as candidate splitting points. To choose between them, we examine how much one data point stands out against the other with respect to its own direction (perpendicular or vertical), by comparing 1) the difference D_p in their perpendicular distances from the straight line connecting the two end points of the segment against 2) the difference D_v in their vertical locations (or y-values), as shown in Figure 2. If $D_p \geq D_v$, we choose P_{PIP} as the split point; otherwise, we choose P_M.

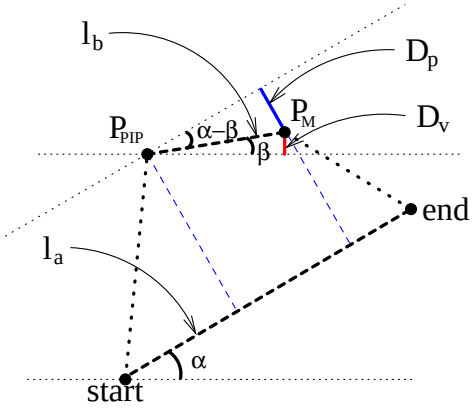


Fig. 2: Relationship between P_{PIP} and P_M

2.3 Decision Module

The Decision Module is responsible for analyzing a segment and making a decision about whether it should be split into two subsegments. In our project, we use 18 local and global attributes and a support vector machine with SMO (Sequential Minimal Optimization)[3] implementation to build the decision module.

2.3.1 Local features

The local feature base is composed of various statistical tests on the segment and other attributes which represent characteristics of the segment. The following are several of the local features that are considered by the SVM in building the decision module for graph segmentation.

a) Correlation Coefficient: A trend can be viewed as a linear relation between the X and Y variables. The Pearson product-moment correlation coefficient measures the tendency of the dependent variable to have a rising or falling linear relationship with the independent variable. It is obtained by dividing the covariance of two random variables X,Y by the product of their standard deviation.

$$r_{XY} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\sqrt{n\sum y_i^2 - (\sum y_i)^2}}$$

The correlation is 1 in the case of an increasing linear relationship, -1 in the case of a decreasing linear relationship,

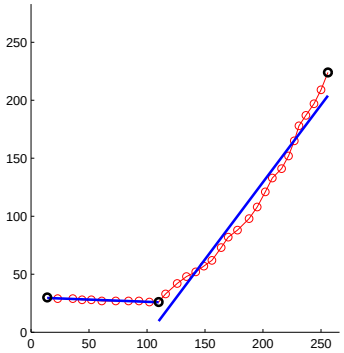


Fig. 3: An example line graph with high correlation coefficient but which should be split into two subsegments, as indicated by the dark circles. The light colored circles are the sampled data points, the three dark circles are the splitting points, and the solid lines are the regression lines for the two subsegments.

and some value in between for all other cases. The closer the coefficient is to either 1 or -1, the stronger the correlation between the variables, thereby suggesting that the segment should not be split. We use the absolute value of the correlation coefficient as a feature in our classifier.

b) Q-test and F-test: Although the correlation coefficient is useful in detecting when a segment should be viewed as a single trend (and thus not split further), it is not sufficient by itself. A flat portion of a line graph may be followed by a very steep rise, resulting in a high correlation coefficient even though the graph should be split into two segments, as in Figure 3. Similarly, a relatively flat but jagged segment will have a low correlation coefficient, even though it should not be split into subsegments, as in the portion of the line graph from 1900 to 1930 in Figure 1.

To address this, we make use of the Q-test[4] and F-test[5], [6] which are measures of change point detection; both test whether a two-segment regression is significantly different from a one-segment regression based on the differences in their respective residuals. The null hypothesis is that no change point has occurred so the two regression models are equal, suggesting that the segment need not be split further into subsegments. The Q-test is specifically designed for this purpose. But as analyzed in [4], the Q-test has less power when the change point is closer to the two endpoints. And according to [7], the Q-test is sensitive to the change in sample size – the larger the sample size, the better its performance. On the other hand, the F-test was designed as a general model fitting test but has been adapted to the two-phase regression problem. The F-test may compensate for problems with the Q-test since it empirically works better when the change point is towards one of the two endpoints.

The Q-test analyzes the likelihoods of the first k data points and the following $n-k$ data points from a Gaussian distribution and takes the logarithm of the likelihood ratio λ:

$$\lambda = \frac{l(k)}{l(n)} = \frac{\hat{\sigma_1}^k \, \hat{\sigma_2}^{n-k}}{\hat{\sigma}^n}$$

where $\hat{\sigma_1}$ and $\hat{\sigma_2}$ are the estimates of the standard errors of the two regression lines, and $\hat{\sigma}$ is the estimate of the standard error of the overall regression line for all data points in the segment. According to Quandt[8], the statistic $-log(\lambda)$ follows the distribution listed in [8].

The F-test statistic is computed as

$$F = \frac{(RSSL - RSS)/2}{RSS/(n-4)}$$

where RSSL is the residual sum of squares of the overall regression line, and RSS is the residual sum of squares of the two-phase piecewise regression lines. The value F here is distributed as an F-distribution with $(2, n-4)$ degrees of freedom as given in [6].

For each sample point P_k in a segment, where $1 < k < n-1$ and n is the total number of data points in the segment, we build the two-phase linear regression models for the data points from P_1 to P_k, and from P_{k+1} to P_n respectively. We consider three significance levels $\alpha = 0.05$, $\alpha = 0.01$, and $\alpha = 0.005$ for the Q-test, and two significance levels $\alpha = 0.1$ and $\alpha = 0.05$ for the F-test in the feature base. Thus we consider a total of five attributes based on the two tests. If a statistic calculated from any k where $1 < k < n-1$ generates a significant result corresponding to one of the significance levels of the Q-test or the F-test, we set the value of the corresponding attribute to 1; otherwise, if no $1 < k < n-1$ in the given segment makes the statistic significant for the corresponding significance level, the attribute is set to 0.

c) Runs Test: Although the two-phase Q-test and F-test may help address the problem of recognizing segments that represent a sequence of two trends, they can give a wrong prediction in some situations where the segment consists of more than two trends. However, computational complexity prevents us from calculating more than a two phase F-test to fit the data points since $O(n^d)$ combination of points would need to be considered for a $d-1$ phase F-test. Failing to reject the null hypothesis for the one-phase regression model doesn't mean the one-phase regression line necessarily fits the data points well since the data points may be better captured by a k-phase regression model where $k > 2$. Therefore, we need a more general statistic to test the goodness of fit between the piecewise linear regression model and the data points.

We make recourse to the Runs Test[9]. The Runs Test detects if a model fits the data points well. For each data point, we calculate its residual from the regression line and categorize it as $+1$ or -1, according to whether the residual is positive or negative. Then the number of *runs* is calculated, where a *run* is a continuous sequence of residuals which belong to the same category, such as consecutive $+1$ or -1. If N_+ is the number of positive residual points and N_- is the number of negative residual points, the mean and standard deviation of the number

of runs suggested by the data points are approximated as

$$R_{mean} = \frac{2\,N_+\,N_-}{N_+ + N_-} + 1$$

$$R_{SD} = \sqrt{\frac{2\,N_+\,N_-\,(2\,N_+\,N_- - N_+ - N_-)}{(N_+ + N_-)^2\,(N_+ + N_- - 1)}}$$

We use the Runs Test to check how well the least squared linear regression for a given segment fits the data points in the segment. If the actual number of *runs* R is larger than $R_{mean} - R_{SD}$, it suggests the least squared linear regression line is a good fit to the segment. We include five features from the Runs Test: the result of the Runs Test, the actual runs R, mean runs R_{mean}, standard deviation of runs R_{SD} for a segment, and the ratio difference between actual runs and mean runs calculated as $|R - R_{mean}|/R_{mean}$.

d) Outlier detection: A line graph may have one or more points that significantly diverge from the overall trend; such points perhaps should be viewed as outliers and not cause a segment to be split further. Thus we employ an outlier detection test based on residuals[10]. To detect the presence of outliers, we assume that the trend can be represented as a regression line; thus all the points within the segment can be represented as $y_i = b + a\,x_i + e_i$ where a and b are calculated from least squared regression. The residual is $e_i = y_i - b - a\,x_i$ and the estimated standard deviation of e_i is

$$s_i = \hat{\sigma}\sqrt{1 - \frac{1}{n} - \frac{(x_i - \bar{x})^2}{\sum (x_i - \bar{x})^2}}$$

where $\hat{\sigma} = \sqrt{\sum e_i^2/(n-2)}$. If $\hat{\sigma}$ equals 0, there are no outliers; otherwise, the standardized residuals $r_i = e_i/s_i$ are computed and $R_m = max|e_i/s_i|$ is used as a test statistic for outlier detection. We use a significance level of $\alpha = 0.01$ and the critical value given in [10]. If R_m is greater than the critical value, outlier detection suggests the presence of an outlier in the sampled data points. If there are multiple r_i that exceed the critical value, then multiple outliers are suggested. We use two features – the result of the outlier test and the number of outliers detected – in our feature base.

e) Other local features: Besides the statistical tests and their corresponding results, other features which help describe the characteristics of the segment are also recorded and passed to the classifier to make a decision. They include:

- Number of data points in the current segment. Hypothesis tests such as the Q-test are sensitive to the number of data points. So this feature is used by the classifier to incorporate the consideration of sample size.
- We hypothesize that the variance in the segment may influence human perception on the split/no-split decision, so we consider the standard deviation of residuals in the segment rescaled by the horizontal length of the segment.
- Standard deviation of the perpendicular distance between the data points and the regression line for the segment, rescaled by the length of the regression line between the two end points of the segment. This feature captures the rescaled deviation existing in the segment from a perpendicular perspective.

2.3.2 Global features

As opposed to other segmentation algorithms which only consider local information obtained solely from the segment under consideration, we also include global features that enable the classifier to consider the individual segment within a larger environment. Our global features include:

- The total number of data points in the whole line graph.
- The relative length of the current segment as a percentage of the whole graph. This feature is included to capture a global view of the segment with respect to the whole graph.

2.3.3 Support Vector Machine as classifier

To produce a training set, each graph in our corpus must be collected from popular media, scanned and sampled, and the ideal segmentation identified by human viewers.[1] These are very time-consuming tasks. Therefore the size of the training set is limited. There are 18 features associated with each training instance, so the feature space is an 18 dimensional space. For our segmentation problem, we chose a support vector machine as classifier because it works very well with high-dimensional data and a relatively small training set and avoids the curse of dimensionality problem[11]. SVM also lessens the chance of overfitting by using the maximum margin separating hyperplane which minimizes the worst-case generalization errors[12], [11]. Furthermore, as opposed to local methods such as nearest neighbor which require locating a small neighborhood for each new test instance, the SVM can build the global hyperplane once from the training set and apply it to test cases with little computation.

The support vector machine provides a maximum margin hyperplane to divide the 18 dimensional space into two parts. In our project, the feature vector is first normalized, and the linear kernel is applied to the feature space to generate a linear hyperplane. We use a linear kernel because an inappropriately chosen degree of polynomial kernel which generates a nonlinear hyperplane might induce overfitting.

We collected a corpus of 234 line graphs and built our training set from this corpus. Each line graph was entered as one instance in the training set along with the appropriate split or no-split decision. In the case of a split decision, each resulting segment is entered as an instance in the training corpus, along with their respective split or no-split decisions, and the process is recursively repeated. This produced a corpus of 649 segments which were used to train our decision module.

3. Examples

Figure 4 displays three examples of segmentations produced by our graph segmentation system. The three line graphs come from three different sources, *USA Today*, *BusinessWeek*, and a

[1] In our message recognition system, a Visual Extraction Module is responsible for using computer vision techniques to analyze an electronic image of a graphic and construct its XML representation, including sampled data points.

local newspaper, and differ from one another with respect to the number of trends and the amount of variance in each trend. The original line graphs are plotted with solid black lines. The split points identified by our segmentation algorithm are shown as circles. The high-level trends are located between each adjacent pair of splitting points, represented by dashed regression lines. We can see from the results that our segmentation algorithm accurately segmented the line graphs with different variances into visually apparent high-level trends.

4. Evaluation

To evaluate our graph segmentation methodology, we performed three evaluation experiments: cross-validation of the learned decision module which is responsible for making a split/no-split decision, and two human subjects experiments which evaluated the entire segmentation algorithm incorporating both the splitting module and the decision module.

4.1 Evaluation of the Decision Module

Since our corpus is small, we use leave-one-out cross validation to test the accuracy of the decision module, where each instance is used once as a test case and all the other 648 instances are used as training cases. The results of all 649 experiments are averaged together to obtain the accuracy of the model. The accuracy obtained from leave-one-out cross validation is 88.3%, compared with the 67.2% accuracy of the baseline decision of no-split (the decision for the majority of instances in the training set). Thus our algorithm has a 31.4% improvement from the baseline.

To identify the importance of the 18 features used by our support vector machine, we applied the recursive feature elimination(RFE) algorithm introduced by Guyon et. al.[13]. It measures the discriminating ability of the attributes by comparing their weights for the corresponding dimension of the hyperplane. It first calculates the weight vector of the SVM which produces the hyperplane that maximizes the margin, and recursively 1) eliminates the feature with the lowest absolute weight in the representation of the hyperplane and 2) rebuilds the hyperplane, until all attributes have been removed one by one. The earlier an attribute is removed, the less discriminating it is and thus the lower its rank. Table 1 lists the features in rank order from most significant to least significant.

Let us examine the top ranked features. The first and fourth features measure the standard deviation in the segment in unit length, either from a perpendicular or a vertical perspective. The rank of these two features indicates that the rescaled standard deviations within a segment play a very important role in making a split decision. The second feature captures global information by measuring the relative length of the current segment; it reveals the fact that when a split/no-split decision is made, features based on local information in the segment are not enough. For identifying trends that are visually apparent to humans, we must consider the segment in a larger context. Although we only have one global feature among the top ten features, it plays an important role in the model, and this global information is ignored by other time series segmentation algorithms. In future work, we will consider other global features, such as the relative location of the segment. In addition to the two standard deviations and the relative length of the segment, the correlation coefficient, Q-test, Runs Test, and number of points in the segment all rank among the top ten features and thus play a significant role in the model.

Rank	Feature name
1	rescaled standard deviation of perpendicular distance
2	relative length of current segment
3	difference between actual runs and mean runs
4	rescaled standard deviation of vertical distance
5	correlation coefficient
6	number of points in current segment
7	Q-test in 995 significance level
8	Q-test in 95 significance level
9	runs test
10	standard deviation of runs
11	outlier detection
12	Q-test in 99 significance level
13	F-test in 95 significance level
14	mean runs
15	total number of points
16	number of outliers
17	actual runs
18	F-test in 90 significance level

Table 1: Features listed in rank order

It is interesting to note that the features coming from the F-test and outlier detection are not ranked in the top ten. This indicates that although the F-test is also a change point detection statistic, it is not as powerful as the Q-test for our segmentation task.

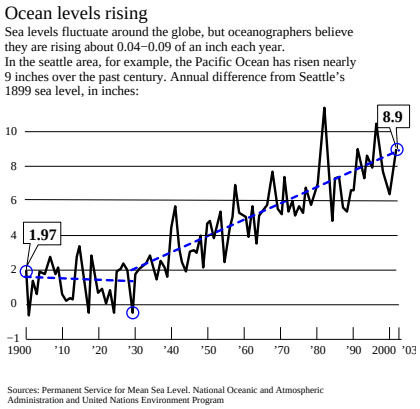
4.2 Evaluation of the entire segmentation algorithm

Recall that our leave-one-out cross validation tested our decision module's ability to make split/no-split decisions on individual segments. To test how well our segmentation algorithm segments entire line graphs into visually apparent trends (both deciding when to split segments and where to split them), we used seven human evaluators. The human subject evaluation had two parts. The first part compared the segmentations produced by our system with those produced by another segmentation method from the literature. The second part was a qualitative evaluation of our graph segmentations. In both experiments, our segmentation for each graph was produced by a model constructed from the other 233 graphs, thus avoiding the problem of biasing the results by including the test graphic in the corpus used to train the model.

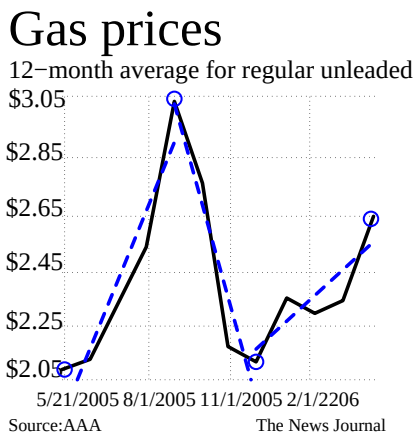
4.2.1 Comparative experiment

In this comparative experiment, seven human evaluators were each given 234 line graphs with two segmentations, one produced by our segmentation algorithm and the other produced by the comparative segmentation algorithm.

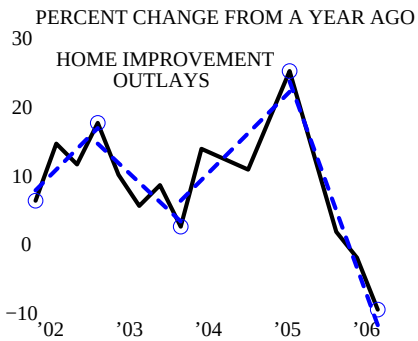
The comparative algorithm reflects existing approaches to time-series segmentation based on error minimization [1] and looks for a predefined k number of segments to minimize the residual sum of squares of the piecewise linear regression. This



(a) A line graph conveying two trends with high variance



(b) A line graph conveying three trends with low variance



(c) A line graph conveying four trends with moderate variance

Fig. 4: Three examples of segmentations produced by our graph segmentation system. The solid lines are the original line graphs, the small circles are the split points, and the dashed lines are the regression lines for the resulting trend segments.

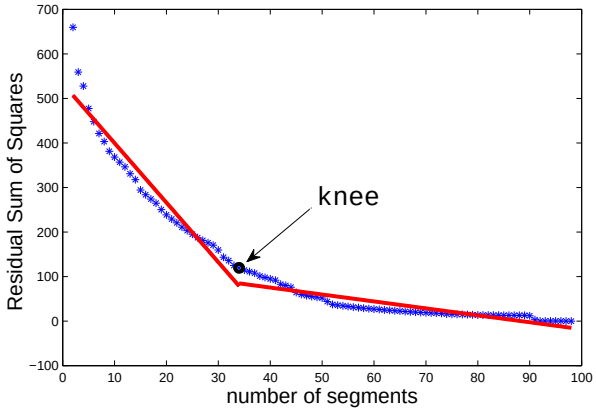


Fig. 5: A plot of segmentation errors against number of segments

error reduction algorithm can be implemented top-down or bottom-up. To be comparable with our top-down segmentation approach, we used the top-down one.

A critical aspect of this method is how to define the number of segments k since the same k for all line graphs is not appropriate. Salvador et. al.[14] suggest a method for identifying the appropriate k for a given line graph by locating the *knee* of the plot of residual sum of squares against the number of segments, as in Figure 5, which is called an evaluation graph and is generated by a top-down iterative process in which one splitting point (i.e. one segment) is added in each iteration. The *knee* is the splitting location which generates a two-phase regression in the evaluation graph and minimizes the residual sum of squares of the evaluation graph.

For each line graph, the human evaluator was given two segmentations, one produced by our segmentation algorithm and one produced by the above comparative algorithm. The order of the appearance of the two segmentations was randomly assigned, to avoid bias resulting from the order of presentation. The evaluators had three options: "The segmentation on the left is better", "The segmentation on the right is better", and "I have no preference".

For 218 of the 234 line graphs, a majority of the evaluators had the same response; on only 16 line graphs was there no majority decision, indicating that these line graphs had differences that made it difficult to identify the better segmentation. For the 218 line graphs where there was a majority decision, the segmentation produced by our system was preferred for 76.1% of the graphs, there was no preference for 8.8% of the graphs, and the segmentation produced by the comparative algorithm was preferred for 15.1% of the graphs. Thus we see that the segmentations produced by our system were preferred five times more often than the segmentations produced by the comparative algorithm.

We also computed how often each evaluator preferred our segmentation, had no preference, or preferred the segmentation produced by the comparative algorithm. All evaluators preferred our segmentation more often than they preferred the other segmentation. Averaging the results for the seven evaluators, we find that our system performed better than or equal to the comparative algorithm 79% of the time.

These results show that our learned graph segmentation algorithm produces better segmentations of line graphs into visually apparent trends than does a traditional algorithm based on error minimization.

4.2.2 Qualitative evaluation of segmentations

Seven human evaluators were given 254 line graphs along with candidate segmentations; 234 of the line graphs were the ones in our corpus with the segmentations produced by our system and 20 were additional line graphs with bad segmentations. The latter were scattered throughout the evaluation set and were included to avoid bias by the evaluators. The evaluators were not told that intentionally bad segmentation examples had been included in the evaluation set. The evaluators were asked to assign a score between 1 and 5 to each segmentation: 5=ideal, 4=very good, 3=acceptable, 2=poor, 1=terrible.

The average rating for the segmentations produced by our

system was 4.25 with 0.55 standard deviation across the 234 line graphs, showing the performance of our segmentation algorithm is between "Very good" and "Ideal". The 20 extra graphs with bad segmentations received an average rating of 1.57 ± 0.44 which is between "Terrible" and "Poor". These results verify that our graph segmentation algorithm successfully segments line graphs into visually apparent trends. This good performance is a result of the learned model generated by our machine learning framework.

5. Related Work

Our line graph segmentation task is related to research on time series segmentation such as [1], [15]. Most of these projects focused on splitting a given time series into a number of segments by finding the piecewise linear approximation or piecewise aggregate approximation which provides the smallest total error or conforms to a maximum error bound within each segment. These research efforts either ask for a fixed number of segments or place a fixed upper bound on errors, which requires prior knowledge about the time series data.

As opposed to the above piecewise approximation approach, another set of time series segmentation research focuses on detecting change point or anomalies in the data. The ARIMA (autoregressive integrated moving average) [16] or ARMA (autoregressive moving average)[17] models fit a formula to the existing data points and predict the range of the incoming data point. Some other regression models[18] fit the time series data using regression methods and calculate the confidence interval of the incoming data point. Change points are identified at those data points outside the predicted upper/lower bound or the defined confidence interval. These methods face the problem of determining a priori how many past data points should be used in their prediction model. Furthermore, the change point detection procedure is very susceptible to the chosen model and distribution.

Similar to our project, some time series segmentation methods[19] rely on statistical tests. However, they use only one or two statistical tests to determine if a segment should be split into subsegments.

These research efforts differ from our work in several ways. They are not concerned with extracting visually identifiable trends but are instead concerned with segmenting based on error minimization, or with pattern detection, prediction, or anomaly detection. In addition, they do not use machine learning to consider a wide variety of features and construct a learned model.

6. Conclusion

This paper has presented our methodology for segmenting a line graph into visually distinguishable trends. Although a number of researchers have addressed the time series segmentation task, to our knowledge our work is the first to attempt to extract trends that are visually apparent to humans, to use both local and global information, and to use machine learning to produce a learned model of graph segmentation. Our leave-one-out cross validation of the module for making a split/no-split decision and our two human subject evaluation experiments of the entire graph segmentation system verify that our graph segmentation algorithm has very good performance. Our graph segmentation algorithm is being incorporated into a Bayesian network that can utilize communicative signals in the line graph (such as a point being annotated with its value) to extract the overall intended messages of line graphs that appear in popular media.

References

[1] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "An online algorithm for segmenting time series," in *Proceedings of IEEE International Conference on Data Mining*, 2001.

[2] F. lai Chung, T.-C. Fu, V. Ng, and R. W. P. Luk, "An evolutionary approach to pattern-based time series segmentation," in *IEEE Transactions on Evolutionary Computation*, vol. 8, October 2004.

[3] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in kernel methods: support vector learning*, 1999, pp. 185–208.

[4] R. E. Quandt, "The estimation of the parameters of a linear regression system obeying two separate regimes," in *Journal of the American Statistical Association*. American Statistical Association, 1958, pp. 873–880.

[5] E. Vieth, "Fitting piecewise linear regression functions to biological responses," in *Journal of Applied Physiology*. American Physiological Society, 1989, pp. 390–396.

[6] R. J. Beckman and R. D. Cook, "Testing for two-phase regressions," in *Technometrics*, 1979, pp. 65–69.

[7] C. A. Diniz and L. C. Brochi, "Robustness of two-phase regression tests," in *REVSTAT - Statistical Journal*, June 2005.

[8] R. E. Quandt, "Tests of the hypothesis that a linear regression system obeys two separate regimes," in *Journal of the American Statistical Association*. American Statistical Association, 1960, pp. 324–330.

[9] D. C. Bradley, G. M. Steil, and R. N. Bergman, "OOPSEG: a data smoothing program for quantitation and isolation of random measurement error," in *Computer Methods and Programs in Biomedicine*, 1995, pp. 67–77.

[10] G. Tietjen, R. Moore, and R. Beckman, "Testing for a single outlier in simple linear regression," in *Technometrics*, 1973, pp. 717–721.

[11] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to Data Mining*. Addison Wesley, 2005.

[12] M. P. S. Brown, W. N. Grundy, D. Lin, N. Cristianini, C. W. Sugnet, T. S. Furey, M. A. Jr., and D. Haussler, "Knowledge based analysis of microarray gene expression data by using support vector machines," in *Proceedings of the National Academy of Sciences*, 2000, pp. 262–267.

[13] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," in *Machine Learning*, 2002, pp. 389–422.

[14] S. Salvador and P. Chan, "Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms," in *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, 2004, pp. 576–584.

[15] E. Terzi and P. Tsaparas, "Efficient algorithms for sequence segmentation," in *SIAM SDM*, 2006.

[16] G. Cellarosi and S. Lodi, "Detecting outbreaks by time series analysis," in *Proceedings of the 15th IEEE symposium on Computer-Based Medical Systems*, 2002, pp. 159–164.

[17] K. Yamanishi and J. ichi Takeuchi, "A unifying framework for detecting outliers and change points from non-stationary time series data," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 676–681.

[18] J. Ma and S. Perkins, "Online novelty detection on temporal sequences," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003, pp. 613–618.

[19] G. P. C. Fung, J. X. Yu, and W. Lam, "News sensitive stock trend prediction," in *Advances in Knowledge Discovery and Data Mining*, 2002, pp. 481–493.