Waveform Shape Recognition for Vertical Profiling with an Ensemble of Support Vector Machines

Greg Brown

December 11, 2009

Abstract

In analyzing computer system performance it is often useful to compare the shapes of waveforms from different system metrics. In this paper, we present a method for the detection of shapes within time series waveforms using an ensemble of support vector machines. By resampling the waveform, detecting shapes independently, and combining the results misclassifications can be filtered out. The shape prediction system outputs probability regions where the system feels that a particular shape will begin and end.

The current results of the shape prediction are quite poor, but some of the intermediate results do suggest that the approach can be successful. We feel that further optimization and refinement of both the SVM classification and combining the ensemble results is needed.

1 Introduction

To understand a computer system's performance, a team of engineers must cope with the dynamic and multilayered nature of the system. A large number of metrics need to be collected on all of the hardware and software layers of the system. The system takes samples of these metrics over time and multiple runs in order to get an accurate view of the system. Then all of the metrics are analyzed by experts who understand how the various aspects of the system are contributing to performance anomalies.

The technique of Vertical Profiling [1] provides a methodology for wading through the large number of performance metrics in order to determine a metric that was causal to a performance anomaly. This analysis compares the shapes of time series waveforms and utilizes expert knowledge of the domain to determine which waveforms are causal to the anomaly waveform. Performing this analysis by hand can take many weeks of work. A learning system that can generate a list of the metrics that could be the root cause of a performance anomaly would greatly reduce the effort needed for doing computer system performance analysis.

We have approached this problem by dividing it into two stages: (1) recognition of shapes within a waveform and (2) determining the causality between a shape in one waveform and the shape of an anomaly in another waveform. In this paper we present a method for the first stage: learning to recognize the shapes within a metric's waveform from a list of possible shapes.

We have decomposed the problem into these two stages to reduce the complexity of the problem and provide more flexibility for the system. Shape recognition is a general classification problem that could be applied to any computer system's metrics. However, differing computer architectures and metrics mean that the causality classification will vary from one system to the next. By subdividing the learning problem only the second stage will need to be relearned for a different computer system.

1.1 Related Work

Hauswirth et al. have already done work on automating the alignment of the metric waveforms in time and determining which metrics correlate with each other [2]. Although this work is critical for aligning metric waveforms in time across multiple runs it does not provide a method for determining the causalities between metrics. Specific shape recognition is necessary for both the user to specify what performance anomaly they are interested in, and to be able to use with expert knowledge of the system since different shapes suggest different behavior of the system.

Finding shapes within waveforms is a common problem in other domains as well such as stock price [3], ECG[4], and EMG[5] analyses. The results in the ECG and EMG areas have 80-90% recall in their problem domains using artificial neural networks.

Syeda-Mahmood, et al. used transformations to compare the shape of a candidate ECG waveform against a database of other waveforms for determining whether the candidate ECG is indicative of a particular heart disorder[4]. This transformation method relies upon the variation from one ECG to another to not vary very widely. Within the ECG domain this is a valid assumption. But for this reason we are not convinced that such an approach would be able to deal with the noise and unstructured data that the metrics are producing. Unlike ECGs where the deviations from the basic shape indicate something being wrong, deviations from the shapes in metrics are expected because the data is inherently noisy. Syeda-Mahmood, et al. cited noise in the input waveforms and variation in measurement equipment as a significant source of their errors, which corroborates this belief.

To learn from the unstructured metric data we use a method from natural language processing of looking at only a single window of the waveform at a time rather than the entire waveform. This avoids the problem of the shape being tied to a particular sample location within the waveform.

1.2 Dataset Description

The set of waveforms used for this research came from the Vertical Profiling experiments[1]. Multiple Java benchmark programs were run on a system and many processor, operating system, and virtual machine metric waveforms were gathered. In this paper we use five of these program runs for a total of 600 waveforms. Each waveform has between 1000 and 5000 data points. Figure 2 depicts a single waveform and some of the shapes that were labeled in that waveform.

2 Shape Learning System

Figure 1 summarizes the shape learning system. The waveforms are first resampled at multiple lower sampling rates. The resampled waveforms (and the original) are processed separately by a support vector machine (SVM). The classifications at each sampling rate are then combined as votes for a shape at a particular point in the waveform. This combination of the results from each SVM helps to remove false classifications, and provides a probability region where a shape could begin and end.

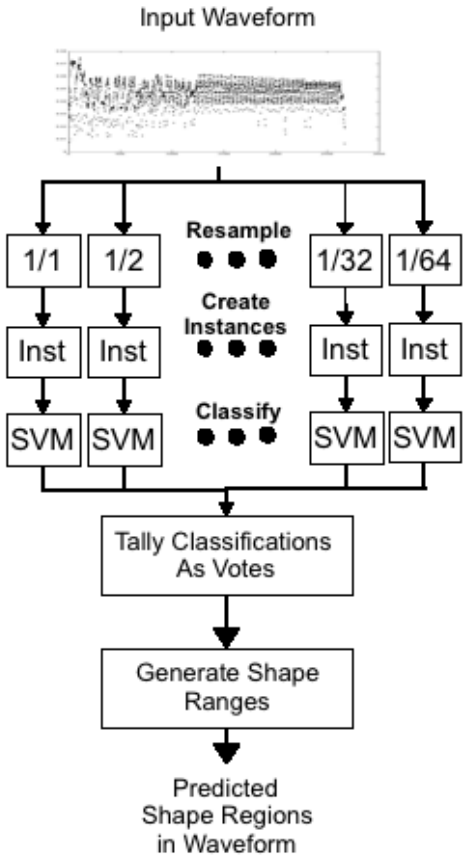

Figure 1: *Diagram of the Shape Learning System*

2.1 Data Preprocessing

The waveforms are resampled using MATLAB's [10] decimate function which applies a 30th order finite impulse response (FIR) filter on the data before resampling. In addition to resampling, this method removes a lot of the noise present in the metric waveforms (see Figure 2). Seven waveforms are created by this resampling at the following rates: 1, 1/2, 1/4, 1/8, 1/16, 1/32, and 1/64. The waveforms are then normalized to the range 0 to 1 based upon the min-max range of the original waveform.

Each of these resampled waveforms is then used to create one instance for each data point in the waveform. Each instance consists of the data point and w samples on either side of the data point. These $2w + 1$ samples constitute a window to look at the waveform through. We also include the first and second differential in this window for a total of $6w$ attributes in each instance. We have found a w of 25 provides sufficient information about the shapes in these waveforms. Larger waveforms would most likely need a different window size.

Each instance will be passed into an SVM and classified as either being a part of a shape or not. The resampling and instance creation process creates $2N$ instances for each input waveform of N samples.

The resampling of the data ensures that both short and long shapes can be detected within the window. We selected the number of resamples so that at the lowest sample rate all of the samples will fit in about two instance windows. This ensures that any length shape up to half the waveform length can be detected.

2.2 Waveform Shapes

The SVMs classify each instance as to whether the midpoint of the window corresponds to being the beginning (begin$< Sh >$) or end (end$< Sh >$) of a particular shape. For example beginGradInc and endGradInc indicate the beginning and ending of a gradual increase in the waveform. An additional class (perPeriodic) indicates the first period in a periodic shape. Table 1 describes all of the shape classes used for the performance metric waveforms. We chose these particular shapes to focus on because they were used in the prior Vertical Profiling research [1]. For different applications other shapes could be used.

These classes are determined in a one vs many manner by training separate SVMs for each class. Each shape has two SVMs (three for Periodic) that determine the location of the shape for a total of 13 SVMs in the system. Each SVM determines whether a particular instance window's midpoint corresponds to that particular class.

2.3 Ensemble of Support Vector Machines

Although a separate SVM is used for each class, we use the same SVM for classifying the instances at all of the sampling rates. However, the results of classifying at each sampling rate are kept separate since they are reclassifying the same location in the waveform. When combining the results the classification of a sample in the original waveform will be determined by a vote from the corresponding sample in each of the resampled waveforms.

The SVMs were implemented using libSVM [11] for the SVM training and classification and Weka [12] for managing and filtering the data instances. The SVMs use a radial basis function of $e^{(-\gamma * |u-v|^2)}$ as their kernel, a gamma of 0.05, and a cost of 100.

Shape Class	Description
beginGradInc	Beginning of a gradual increase in the waveform.
endGradInc	End of a gradual increase of the waveform.
beginGradDec	Beginning of a gradual decrease in the waveform.
endGradDec	End of a gradual decrease of the waveform.
beginStepUp	Beginning of a step increase in the waveform.
endStepUp	End of a step increase of the waveform.
beginStepDown	Beginning of a step decrease in the waveform.
endStepDown	End of a step decrease of the waveform.
beginBurst	Beginning of a burst or distortion in the waveform.
endBurst	End of a burst or distortion of the waveform.
beginPeriodic	Beginning of a periodic shape in the waveform.
endPeriodic	End of a periodic shape in the waveform.
perPeriodic	At the end of the first period in a periodic shape.

Table 1: *List of Shape Classes*

2.4 Combining the Ensemble Classifications

After classification with the SVMs, each sample of the original waveform now has 6 classifications per class. These classifications need to be combined into a single, coherent prediction for each sample. The SVMs will necessarily have a good number of misclassifications for three reasons: 1) errors by the classifiers due to noise, 2) loss of information as the waveforms are resampled, and 3) the limited instance window size.

Additionally, the starting and ending points of all of these shapes are open to interpretation and the labeled dataset we have reflects that ambiguity. Although the beginning and ending of a shape are assigned to a single data sample the couple of samples before and after each are also valid as start and end points. To allow for this ambiguity the predictions are combined to form ranges within which a class occurs. Figure 3 shows an example of the prediction ranges compared to the labels for an example waveform.

For each shape class the 7 classifications for each

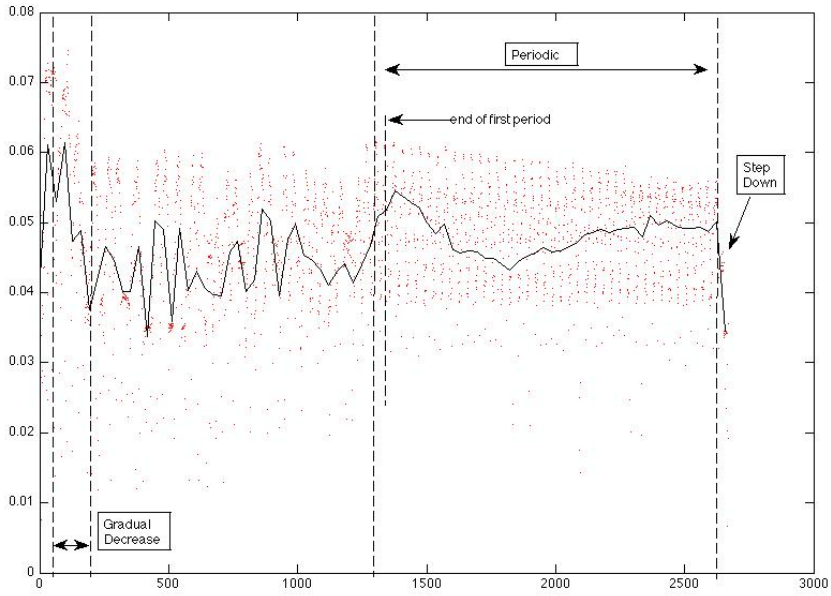

Figure 2: *An example metric waveform and the waveform resampled at 1/32. Some of the labeled shapes in the waveform are annotated.*

sample are summed as votes. Ranges are then determined by scanning through these vote totals one sample at a time. The start of a range is triggered when the vote total is above one. When the vote total falls back to zero the range is complete. Each range constitutes a probability density where a particular shape class is likely to occur. Currently we are not making use of this probability density but it may be useful as an input in future work.

Final probability ranges for begin$< Sh >$, end$< Sh >$, and per$< Sh >$ classes are all generated with the above method. We do not attempt to filter the ranges so that there are an equal number of begin and end classes. Because multiple shapes could be overlapping in a single waveform (a long term gradual increase could be composed of multiple shorter gradual increases) there is not necessarily a one to one relationship between the number of class ranges.

3 Experiment Design

I labeled the waveforms by hand based upon which shapes were apparent when looking at the overall waveform. On average about one minute was spent labeling each of the 600 waveforms. It is quite likely that there are more shapes present than what has actually been labeled which means that there will be a decent number of misclassifications due to the labeling of the data being incorrect. For instance there are likely to be periodic shapes in the waveforms that I did not recognize when labeling the waveforms.

We then divided these 600 waveforms into a number of separate datasets for our experiments. First 43 waveforms that all came from a single benchmark program were separated out to be used as an "independent testset". The remaining 557 waveforms were randomly distributed into the other datasets: 400 for training, 50 for validation, and 50 for testing. The remaining 7 waveforms were not used.

Table 2 summarizes how many positive examples of each shape are in each dataset. The following subsections describe how these datasets were used.

3.1 Training

We generated instances from the trainingset of 400 waveforms as discussed in section 2.1. The SVM for each class was trained on instances from all of the resamplings of each waveform. This yields between 620 and 3200 positive examples of each class for an SVM to learn from, and on average 1.5 million negative examples. Training with such a large discrepancy results in very poor recall of the

Shape Name	Training Set	Validation Set	Random Testset	Independent Testset
GradInc	119	18	16	17
GradDec	72	10	7	16
StepUp	217	21	29	19
StepDown	202	28	32	19
Burst	364	53	47	6
Periodic	253	33	33	32
Total Waveforms	400	50	50	43

Table 2: *Summary breakdown of each dataset.*

positive examples of the classes, as well as taking a very long time to train all of the SVMs.

To alleviate this problem, we used a reduced subset of the instances for training. The subset consists of all of the positive examples and a random sampling of the negative examples. Negative instances are added to the subset until the ratio of positive to negative instances is 1:19. This method does drastically change the ratio of positive to negative instances away from their "natural" occurrence rate which does result in increasing the misclassification of negative instances. And as the number of negative instances in the training set is reduced we do see a decrease in the percentage of incorrectly classified negative examples. We chose the above ratio by experimentally trying a number of ratios on the validation set. The ratio of 1:19 gave fairly good results while not taking too long to train.

3.2 Validation Dataset

We used the validation dataset to allow testing and development of this learning system. Specifically we used this dataset to optimize and test the parameters of the system.

For instance, the initial experiments did not have nearly enough positive examples and performed very poorly on the validation dataset. This led us to expand the size of the training set from 100 to 400 and use all of the resamplings to train each SVM.

The validation set was also used to determine the size of the waveform window, whether the first and second differentials were helpful in the classification, the optimal gamma and cost parameters to use in training, and the ratio of positive to negative examples in the training set.

For most of these experiments we worked exclusively on the beginGradInc class and then used these same parameters on the other classes. The parameter optimizations certainly do not translate perfectly to the other classes, and this is an area that needs to be improved in the future.

3.3 Testing Dataset

The two testing data sets ("random testset" and the "independent testset") were only used for generating the final results of our experiment. Using a test set distinct from the validation dataset minimizes the extent that we were tuning our system to perform well on our test dataset. The use of a completely independent dataset further ensures that the results can be replicated on other waveforms in the future. The independent test set is composed of all the waveforms from a single program run. No waveforms from this particular program were used in any of our other datasets.

4 Results

Table 3 gives a summary of the results on the random and independent testsets. The results are given as the ability of the classifiers to generate ranges which the begin$< Sh >$ and end$< Sh >$ classes fall within. For a given class, we evaluate positive and negative regions. A positive region is a continuous set of samples where the system has predicted that the class will occur. A negative region is any continuous set of samples between the positive regions. These regions are compared against where the class label is located. The results are determined to be true positive(TP), true negative(TN), false positive(FP), and false negative(FN) according to the following rules for each class C:

- TP: positive region contains a sample labeled with class C.
- FP: positive region does not contains a sample labeled with class C.
- TN: negative region does not contains a sample labeled with class C.
- FN: negative region contains a sample labeled with class C.

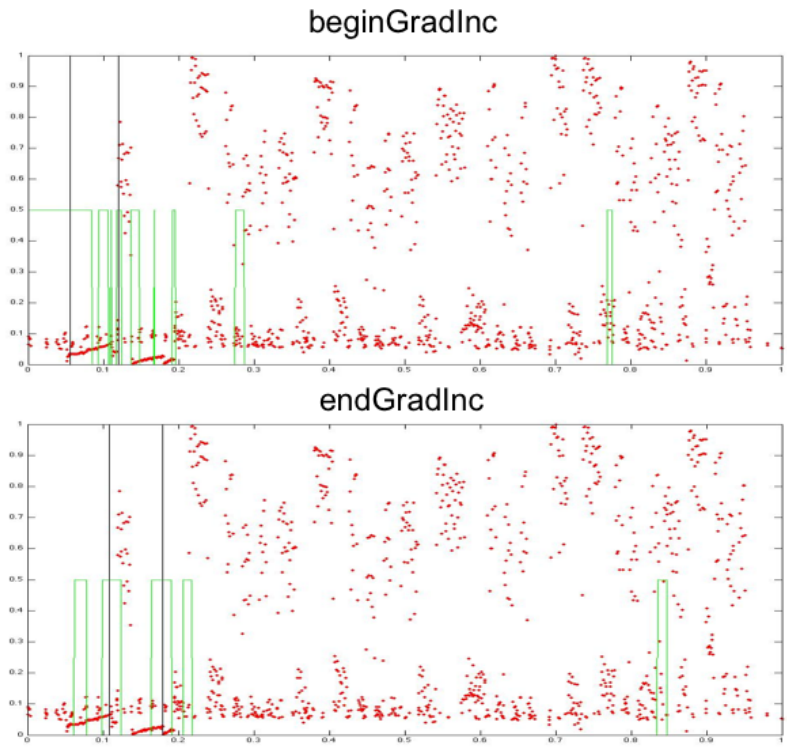

Figure 3: *Predicted regions (outlined by green lines) where the beginGradInc (top) and endGradInc(bottom) classes occur for an example waveform (red dots). Class labels are indicated by the vertical black lines.*

Class	**Random Testset**		Confusion Matrix		**Independent Testset**		Confusion Matrix	
	Recall	Precision	TP FP	FN TN	Recall	Precision	TP FP	FN TN
beginGradInc	50%	4.4%	8 172	8 222	23%	2.0%	4 194	13 229
endGradInc	13%	1.3%	2 154	14 193	0.0%	0.0%	0 121	17 151
beginGradDec	29%	1.2%	2 171	5 220	13%	0.9%	2 222	14 256
endGradDec	0%	0.0%	0 107	7 152	0%	0.0%	0 91	16 123
beginStepUp	48%	7.7%	14 168	15 218	0%	0.0%	0 185	19 212
endStepUp	62%	12.9%	18 121	11 178	21%	2.4%	4 161	15 196
beginStepDown	47%	8.7%	15 157	17 206	16%	1.7%	3 171	16 203
endStepDown	31%	5.9%	10 160	22 201	0%	0.0%	0 157	19 183
beginBurst	21%	5.8%	10 162	37 196	0%	0.0%	0 117	6 155
endBurst	40%	10.6%	19 160	28 208	0%	0.0%	0 128	6 167

Table 3: *Final Ensemble results for shape class recall and precision on the random and independent testsets.*

These metrics are then used to calculate the overall recall and precision metrics for each class as shown in Equations 1 and 2.

$$Recall = \frac{TP}{TP + FN} \quad (1)$$

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

These metrics are not ideal since the size of the individual regions varies widely and a very good result could be achieved simply by predicting that the positive region encompasses the entire waveform. For the moment though no other metrics do a very good job of handling the case where there are multiple shapes in a waveform that are overlapping one another. A confusion matrix of the TP, FP, TN, and FN raw results is also included in Table 3 to show that there are many negative regions being predicted in the waveform as well.

NOTE: Periodic shape results were not available in time for this paper.

5 Discussion

Our results are currently not very good. However there are indications that the method being pursued has some promise and that improvements to the system can be made. On the random testset the beginGradInc and both StepUp classes are having some success at predicting the start and end of shapes. Most of the time that we spent on optimizing the system focused on the beginGradInc class. The results of that optimization were just blindly applied to all of the other classes. This suggests that optimizing the other classes could improve the results significantly.

The precision of the prediction is very poor across all classes. This is somewhat indicative of the ambiguity in what a person considers to be a shape. For instance, in figure 3 there were two gradual increases that were labeled. However, a different person could also have labeled the entirety of the beginning of the waveform as a gradual increase. The predicted regions in this figure suggest that the learning system felt that there was potentially more than one shape occurring during the beginning of this waveform. We find this modest amount of success heartening although there is clearly a lot of work to do refining the shape classification system.

The results were even worse on the independent testset than on the random testset. This is not surprising given that the random testset is not independent from the waveforms that the system trained on. But it is also clear that those classes with higher recall on the random testset got better results on the independent testset.

6 Future Work

There are a number of avenues that may allow further improvements to the shape classification system. As previously mentioned, the SVM parameters have largely been optimized with a focus on improving the GradInc class results (and especially on just the beginGradInc class). Performing experiments to optimize the parameters and features for the other classes would likely improve their results.

There are also about 700 unlabeled waveforms available, and the additional examples they provide might be able to improve the classification results.

Additionally it would be beneficial to relabel the existing dataset and to have multiple other people label the waveforms. This would help determine how much ambiguity exists in the waveforms and give a wider range of results to compare against. At the very least the testsets need to be labeled multiple people so that there is less ambiguity as to how many different shapes are present in them.

The SVM parameters for the above results were chosen to minimize the number of misclassifications generated at the expense of recalling the positive examples. Our idea at the time was to limit how many false positives we got so that they would not swamp out the good data. This also allowed the combining of the results from the SVMs to be far simpler since it did not have to filter many misclassifications. We believe this was actually a poor choice in the end. When run on the random testset the resulting SVMs only achieved 15-20% recall of the positive examples before they were combined as votes. Although the ensemble method used did greatly improve these results, a much higher initial recall may have been able to get significantly better results.

7 Conclusion

This paper has presented a novel method for shape detection within a time series waveform using an ensemble of support vector machines. Shape detection in waveforms is a common problem in a number of problem domains. This paper specifically looked at the analysis of computer system performance by finding shapes in metric waveforms.

Although our results at this time are quite poor, we believe that further refinement to the system may be able to achieve results good enough for

determining the causal relationship between computer system metric waveforms.

References

[1] Hauswirth et al. *Vertical profiling: understanding the behavior of object-oriented applications*. Proceedings of the 19th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications (2004) pp. 251-269

[2] Hauswirth et al. *Automating vertical profiling*. Proceedings of the 20th annual ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications (2005) pp. 281-296

[3] Kainijo and Tanigawa. *Stock price pattern recognition - a recurrent neural network approach*. cse.ust.hk

[4] Syeda-Mahmood et al. *Shape-based matching of ECG recordings*. Engineering in medicine and biology society (2007)

[5] Subasi et al. *Classification of EMG signals using wavelet neural network*. Journal of neuroscience methods (2006)

[6] Roddick and Spiliopoulou. *A survey of temporal knowledge discovery paradigms and methods*. IEEE Transactions on Knowledge and Data Engineering (2002)

[7] Agrawal et al. *Querying shapes of histories*. Proceedings of the 21st VLDB (1995)

[8] Guralnik and Srivastava. *Event detection from time series data*. Conference on Knowledge discovery and data mining (1999)

[9] Antunes and Oliveira. *Temporal data mining: An overview*. KDD Workshop on Temporal Data Mining (2001)

[10] *MATLAB* version 7.8.0.347. Natick, Massachusetts: The MathWorks Inc., 2009.

[11] Chih-Chung Chang and Chih-Jen Lin, *LIBSVM : a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm

[12] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); *The WEKA Data Mining Software: An Update*; SIGKDD Explorations, Volume 11, Issue 1.