Chapter 10

Conclusion

10.1 Discussion

Question 1: *Usually a dynamic system has delays and feedback. Can OMEGA handle systems with infinite delays, and with elastic delays?*

OMEGA handles those systems with finite orders of delays. A system with elastic delays means that the order of delay varies from time to time. OMEGA is applicable to systems with elastic delays, if we know the range of the delays. We can assign the maximum order of the elastic delays to be the order for OMEGA. However, notice that with redundant order of delays, OMEGA may perform inefficiently.

Question 2: *Both OMEGA and Hidden Markov Models can handle time series. When should we use OMEGA instead of HMM?*

Some systems have hidden states, and the observable input and/or output of the systems are the manifestation of the hidden states. Hidden Markov Models are good at modeling the hidden states and their transition relationships. Conversely, OMEGA analyzes the complicated distribution of the input and output. To some extent, OMEGA may be capable of handling systems with hidden states. However, in case there are no hidden states, OMEGA will perform better.

For example, in the driving domain, because of different road conditions and traffic conditions, the distribution of the input and output tends to be very complicated. However, it seems to us that probably there are not too many hidden states standing between the input and output. Therefore, OMEGA may be better for the driving domain.

Question 3: *Neural Networks, especially Recurrent Networks, are often used for forecasting. Can Neural Networks be used to do system classification? If so, what is the advantage of OMEGA compared with Neural Networks?*

As we mention in Section 2.5., we decompose $P((x_i, y_i) \mid S_p)$ into the product of $P(x_i \mid S_p)$ and $P(y_i \mid S_p, x_i)$. We can use any machine learning method to approximate $P(y_i \mid S_p, x_i)$. Hence, Neural Networks can also be used to do system classification.

However, for each candidate system S_p, we should prepare a Neural Network. If we have 10,000 candidate systems, and the time series to be classified is 40,000 units long ($i = 1, ..., 40,000$), then we will have to try all of the candidate system at every time step. The computational cost will be 4 million units, which is not desirable. However, as a memory-based learning method, OMEGA can focus on the promising candidate systems from the very beginning. In this sense, OMEGA is cheaper than any parametric machine learning methods, such as Neural Networks.

Question 4: *What about the computational efficiency of OMEGA compared with HMM, global linear model ARMA, as well as Neural Networks?*

Concerning computational complexity, the training cost of the global linear model, ARMA, is $O(M^3 + T)$, where T is the total length of training time series samples, while M is decided by the model size of ARMA[1]. The training cost of HMM is $O(N^2 \times T^2)$, where N is the number

1. An ARMA model consists of two parts: AutoRegression (AR) and Moving Average (MA). If the window size of AutoRegression is p, and the window size of Moving Average is q, then $M = max(p, q + 1)$.

of hidden states in the HMM. Typically, the training process of a Neural Network is divided into several epochs. If there are W weights in a Neural Network, each epoch takes $O(W \times T)$. However, the worst-case number of epochs can be exponential in d, which is the number of input attributes. OMEGA does not need any training process, but it re-organizes the memory of training time series data points in the form of a kd-tree, which takes $O(d^2 \times T + T \times \log T)$.

To evaluate a time series query, the ARMA approach is to estimate the parameters of the ARMA model. Hence, the computational cost of evaluating a time series query is similar to the training cost. If the length of a time series query is t, the computation cost of evaluating is $O(M^3 + t)$. The evaluating cost in HMM model is $O(N \times t)$, while that of a Neural Network is $O(W \times t)$. The order of computation complexity in OMEGA is also proportional to t, but in addition depends on what machine learning method is used to approximate $P(X_{qi} | S_p)$ and $P(y_{qi} | S_p, X_{qi})$. For example, if OMEGA uses locally weighted logistic regression as the approximator, the computational cost is $O((d^3 + d \times T) \times t)$, where T is the number of training data points. However, with the help of cached kd-tree, the cost can be greatly reduced if the dimensionality, d, is not too large.

In summary, unlike HMMs and Neural Networks, OMEGA is not expensive to train. However, evaluating a time series query in OMEGA is not trivial in computation. Based-on our empirical knowledge, OMEGA[2] is still fast enough to be an on-line system classifier. Also notice that ARMA, HMMs and Neural Networks have to try all candidate systems at every time step of a time series query, hence if there are S candidate systems, their computational cost of evaluating a time series query are $O((M^3 + t) \times S)$, $O(N \times t \times S)$ and $O(W \times t \times S)$ respectively. On the other hand, OMEGA can quickly focus on the promising candidate systems at the beginning of the query, so that its cost is $O((d^3 + d \times T) \times t \times s)$, sometimes $s << S$.

2. In our experiments, we used locally weighted logistic regression as the approximator of $P(X_{qi} | S_p)$ and $P(y_{qi} | S_p, X_{qi})$.

Question 5: *Ideally OMEGA assumes the input and output are fully observable and the output is fully determined by the input. However, in practice, this assumption is often violated. How badly will OMEGA perform when the assumption is violated?*

OMEGA studies the mapping between the input distribution and the output distribution. If there are some patterns in the mapping which can be used to distinguish different systems, OMEGA will work well, no matter whether or not the input and output are fully observable.

Question 6: *The principle of OMEGA is to calculate the residuals between the predictions and the observed results, then summarize the residuals in the form of likelihood. This is similar to Kalman filter. What is the difference between OMEGA and Kalman filters?*

Kalman filters assume that we know the closed-form formula for a system, and its goal is to estimate the parameters of the formula by minimizing the residuals between the predictions and the observations. The Kalman filter approach can be modified to do system classification, if we know the closed-form formula of the system. However in many cases we do not explicitly know the mechanism of the system, so we cannot go through the mathematical process of Kalman filters. OMEGA is a non-parametric method, which regards the system as a black box. This is the main difference between OMEGA and Kalman filters.

Question 7: *What makes some people's tennis styles similar? Is there any way to learn the similarity of individual styles directly?*

For the tennis experiment, the only instruction that we gave to the participants was: "hit the ball to make it move across the net." Based on our observation of the tennis experiment, the right-handed people are more likely to hit the ball toward the top-left corner of the court, while some left-handed people tend to make the ball move to the top-middle or top-right. Some people hit the ball harder than others, some people hit the ball once it comes across the net, etc. All the above are relevant to individual tennis styles.

Can we use some simple statistical features to do the system classification, such as the mean values of the contact angle, speed, the position of the contact? It is possible that the simple features work in some cases. However, in those cases, the input variables, i.e. the serving variables, must be uniformly distributed, because the contact angle, speed and the position of the contact, are also dependent on the input variables. OMEGA is more powerful than the feature approach since OMEGA studies the mapping between the input distribution and output distribution.

Question 8: *Suppose OMEGA is employed to detect several drivers' sobriety conditions. Each driver has both "sober" training data sets and "drunk" training data sets. Certainly we can use each driver's two kinds of training data sets to detect his sobriety. But is it helpful to put every driver's sober training data sets together as a mega sober training data set?*

Since all alert drivers share some common behavior, it is helpful to collect all "sober" training datasets into a big pool. However, for different drivers, the definition of being alert may be different. A cowboy's alert action may look very wild to a conservative person. Thus, if possible, a better idea is to put the training dataset generated by the *same type* of people together.

Question 9: *Is OMEGA good for speech?*

Because of accent and emphasis, the same sentence may be pronounced in different ways. In other words, for the same sentence, the distribution of the signal may be different, but the hidden states are always the same. Referring to the answer to Question 2, OMEGA is not good at approximating the relationship among the hidden states, but focuses on the distribution pattern of the signals. Therefore, in our point of view, OMEGA is not good for speech recognition.

10.2 Contributions

In this thesis, we explore a coherent framework to detect the underlying system that produced a given sequence of data points. This set of data points can be a time series in which the order of the sequence is important, or it can be a non-time series as well. Our approach is to transform the time series or non-time series into a set of data points with low input dimensionality, then use efficient memory information retrieval techniques and machine learning methods to do a series of classifications, and employ likelihood analysis and hypothesis testing to summarize the classification results as the final detection conclusion. The framework of our system is illustrated in Figure 10-1. The original contributions of this work are:

1. To our best knowledge, our work, for the first time in the literature, employs state-of-art data mining techniques in conjunction with memory-based learning methods to approach time series detection problem. Compared with other alternative methods, our method is simple to understand and easy to implement, it is robust for different types of systems with noisy training data points, it is adaptive when the density and the noise level of the training data points vary in different regions, it is flexible because it does not request fixed thresholds to distinguish various categories, it is efficient not only because it is capable of processing the classification quickly but also can it focus on the promising categories from the very beginning, and based on our empirical evaluation, it is more accurate than other methods.

2. We combined the locally weighted paradigm with logistic regression to be a new memory-based classification methods. Unlike the other memory-based classifiers, it is capable of extrapolating as well as interpolating. It is competent in accuracy, and with some extra useful features, especially, confidence interval. With the help of cached kd-tree, it is a very efficient classification method.

3. As known for many years, kd-tree can be used to re-organize the memory so as to retrieve

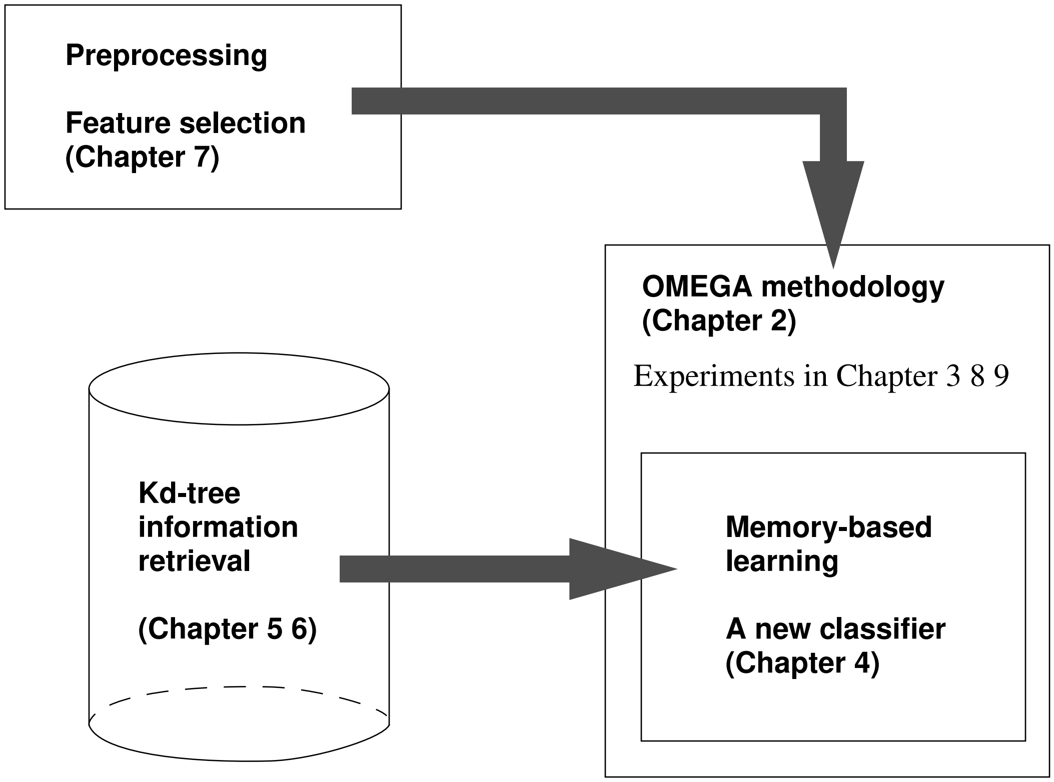

Figure 10-1: The structure of OMEGA system and the organization of the thesis.

the useful information efficiently. By caching well-selected information into the kd-tree's node, we found a way to dramatically improve the efficiency of memory-based learning methods, including Kernel regression, locally weighted linear regression, and locally weighted logistic regression. Recently, cached kd-trees have also been applied to improve the efficiency of EM clustering [Moore, 98].

4. Due to the progress in improving the efficiency of the variety of learning methods, intensive cross-validation becomes feasible. We used intensive cross-validation to do feature selection, especially we explored several greedy algorithms to perform the selecting even faster while without severe loss in precision. We tried applying these algorithms to select the useful features so as to recognize Chinese handwriting off-line. Our prototype showed the accuracy could be over 95%.

10.3 Future research

1. Referring to Figure 10-1, the pre-processing module is to transform a time series into a set of data points in which the time order is no longer important, and to reduce the dimensionality of the dataset. Although in our system, we employ Principal Component Analysis and Feature Selection to reduce the dimensionality, there is no guarantee that we achieve our goal in any domain. In case the memory data points distribute in clusters, [Agrawal et al, 98] may be worth trying. More research should be done to attack the curse of dimensionality.

   One promising solution is that we can approximate the relationship among the input attribues using Bayesian network [Pearl, 88] or dynamic Bayesian network [Dean et al, 88]. By learning the configuration and the transition probabilities of the Bayes net [Heckerman et al, 95], we can compress each data points from high-dimensional space into a lower one, even a scalar [Frey, 98] [Davis, 98].

2. In this thesis research, we treated the system as a blackbox, we only study its inputs and outputs. This is desirable for many domains, because sometimes we do not have the precise domain knowledge. However, sometimes we *do* know somethings about the internal structure of the system, then we should exploit this knowledge because it is helpful to enhance the detection accuracy. [Heckerman, 96] and many other papers suggest that Bayesian network is capable of being a good system approximator with many advantageous properties.

   We propose that by using a same Bayes network, we can get double benefits: improving the accuracy, as well as reducing the dimensionality so as to improve the computational efficiency.

10.4 Applications

There are many possible applications, listed in Chapter 1. In this section, we discuss three applications in further depth.

Financial modeling

The importance of financial modeling is obvious: it helps to gain profit from the stock market, and avoid bad investments, such as the recent failure of Long Term Capital Management (LTCM) hedge fund. There are many researchers doing financial modeling, including some Nobel Prize winners. Why should we compete with them?

Most financial models assume the behavior of the financial market is controlled by a unique mechanism. Most Wall Street researchers want to make this unique model more complicated in order to fit all possible scenarios in the financial world. In contrary, we believe that although the stock index, like S&P index, is only an one-dimensional time series, the underlying mechanism of the financial market is not unique, instead, there are several different underlying control systems either working at the same time or switching from time to time. Suppose given the recent behavior of the stock market, including the various influencing factors like Fed's interests, we can use OMEGA to retrieve the similar historic data clips from the database, figure out which underlying mechanism is working nowadays. And based on that, we can predict what will happen to the stock market, with a certain confidence measurement.

Web server monitoring

The rapid growth of internet has greatly increased the pressure on administrators to quickly detect and resolve service problems. Typically, the detection job is done either by some ad hoc models to estimate weekly patterns [Maxion, 90], or by specifying threshold testing [Hellerstein et al, 98].

Using the techniques explored in this thesis, we are capable of detecting more complicated patterns efficiently, and distinguishing the patterns by specifying thresholds which are adaptable to datasets with different distribution densities and different noise levels. In other words, our technique may be more robust and accurate than the previous approaches.

Embedded detection device

To monitor if an engine works normally, we can embed a chip into the engine so that whenever it runs, the chip takes records of the engine's signals. If one day, the operator finds "sometimes" the engine did not work normally, he can pull out the chip from the engine, insert it into a device hooked to his home PC. His home PC is linked to a super server somewhere else through the internet. By comparing this engine's signal time series with those in the super server's database, the server can tell the operator when and how his engine went wrong. Thus, it is more convenient for the operator to decide if the engine needs repairing.

Compared with the conventional methods, which are based on the domain knowledge, our approach has more advantages: (1) Since there are so many engine nowadays in the world, and they are updated so quickly, it is not very convenient to update the conventional diagnosis system, because usually they are installed in the engines. For our distributed system, we can simply update the knowledge in our central super server, we do not need to modify the product we have sold to our customers one by one. (2) The conventional methods are of "we design, you use" style, our approach can interactively collect new data from the customers, then learn from them. Hence, with more and more experience, our system can automatically become more intelligent.