ACTIVE LEZI: AN INCREMENTAL PARSING ALGORITHM FOR SEQUENTIAL PREDICTION

KARTHIK GOPALRATNAM and DIANE J. COOK

Department of Computer Science and Engineering
Box 19015, 416 Yates
The University of Texas at Arlington, Arlington, Texas 76019-0015
{gopalara, cook} @cse.uta.edu

Received July 18, 2003
Revised September 16, 2004

Prediction is an important component in a variety of domains in Artificial Intelligence and Machine Learning, in order that Intelligent Systems may make more informed and reliable decisions. Certain domains require that prediction be performed on sequences of events that can typically be modeled as stochastic processes. This work presents *Active LeZi (ALZ)*, a sequential prediction algorithm that is founded on an Information Theoretic approach, and is based on the acclaimed LZ78 family of data compression algorithms. The efficacy of this algorithm in a typical Smart Environment – the Smart Home, is demonstrated by employing this algorithm to predict device usage in the home. The performance of this algorithm is tested on synthetic data sets that are representative of typical interactions between a Smart Home and the inhabitant. In addition, for the Smart Home environment, we introduce a method of learning a measure of the relative time between actions using ALZ, and demonstrate the efficacy of this approach on synthetic Smart Home data.

Keywords: Sequential Prediction, Data Compression, Markov Models, Prediction by Partial Match.

1. Introduction

As Intelligent Systems become more common and diversified, an important capability that they need to possess is the ability to predict the occurrence of various events in order to be able to adapt and have the versatility to make decisions in a variety of situations. Especially common is the problem of sequential prediction – given a sequence of events, how do we predict the next event based on a limited known history.

In this work, we present *Active LeZi*, a prediction algorithm that approaches this prediction problem from an information theoretic standpoint. For any sequence of events that can be modeled as a stochastic process, this algorithm employs the power of Markov models to optimally predict the next symbol in any stochastic sequence. Many situations demand that the prediction algorithm be capable of incrementally gathering information and deliver real time, "online" predictions. *Active LeZi* is based on the LZ78 data compression algorithm, which employs incremental parsing, thereby addressing this requirement.

Consider a sequence of events being generated by an arbitrary deterministic source, which can be represented by the stochastic process $X = \{x_i\}$. The sequential prediction problem can then be stated as follows. Given the sequence of symbols $\{x_1, x_2, \ldots x_i\}$, what is the next symbol x_{i+1}? Well-investigated text compression methods have established that good compression algorithms are also good predictors. According to Information Theory, a predictor that builds a model whose entropy approaches that of the source achieves greater predictive accuracy. Also, it has been shown that a predictor with an order that grows at a rate approximating the entropy rate of the source is an optimal predictor. Another motivation to look to the field of text compression is that such algorithms are essentially incremental parsing algorithms, which is an extremely desirable quality in our search for an online predictor. *Active LeZi* is a predictor addressing this prediction problem, and is based on the above-mentioned motivations.

2. Related Work

The problem of constructing a universal predictor for sequential prediction of arbitrary deterministic sequences was first considered in (Ref. 1), where the existence of universal predictors that could optimally predict the next of *any* deterministic sequence was proved. They also defined the concept of *predictability*, which is central to the idea of sequential prediction, and proved that Markov predictors based on the LZ78 family of compression algorithms attained optimal predictability.

These concepts were implemented in branch prediction in computer programs (Ref. 2), and page pre-fetching into memory (Ref. 3). Another predictive method based on the LZ78 algorithm was developed by Bhattacharya et. al. for a predictive framework for mobility tracking in PCS networks (Ref. 4).

3. Predictability and Finite State Predictors

Consider a stochastic sequence $x_1^n = x_1, x_2, \ldots\ldots x_n$. At time t, the predictor will have to predict what the next symbol x_t is going to be, based on the past history – the sequence of input symbols $x_1^{t-1} = x_1, x_2, \ldots\ldots x_{t-1}$. Let the predicted symbol be b_t. There is a loss function $l(b_t, x_t)$ associated with every such prediction, and the object of any predictor is to minimize the fraction of prediction errors, π associated with the predictor - i.e., the quantity:

$$\pi = \frac{1}{n}\sum_{t=1}^{n} l(b_t, x_t) \qquad (1)$$

must be minimized (Ref. 1).

Given the resources in a practical situation, the predictor that is capable of possibly meeting these requirements must be a member of the set of all possible finite state machines (FSM's). Consider the set of all possible finite state predictors with S states. Then the *S-state predictability* of the sequence x^n (denoted by $\pi_S(x^n)$), is defined as the minimum fraction of prediction errors made by an FS predictor with S-states. This is a measure of the performance of the best possible predictor with S states, with reference to a given sequence. For a fixed-length sequence, as S is increased, the best possible predictor for that sequence will eventually make zero errors. The *finite state predictability* for a particular sequence is then defined as the S – state predictability for very large S, and very large n, i.e. the finite state predictability of a particular sequence is

$$\lim_{S\to\infty}\lim_{n\to\infty}\pi_S(x^n).$$

FS predictability is an indicator of the best possible sequential prediction that can be made on an arbitrarily long sequence of input symbols by any FSM. This quantity is analogous to FS compressibility, as defined in (Ref. 5), where a value of zero for the FS predictability indicates perfect predictability and a value of ½ indicates perfect unpredictability.

This notion of predictability enables a different optimal FS predictor for every individual sequence, but it has been shown in (Ref. 1) that there exist *universal* FS predictors that, independent of the particular sequence being considered, always attain the FS predictability.

3.1 *Prediction using Incremental Parsing – The LZ78 algorithm*

An important result that is derived in (Ref. 1) is that a sub-class of the class of FS predictors, the class of Markov Predictors, performs asymptotically as well as *any* FSM. That is, a sequential Markov predictor whose order varies at the appropriate rate with respect to the number of symbols seen in the sequence will eventually attain FS predictability.

The LZ78 data compression algorithm as suggested by Lempel and Ziv (Ref. 5), is an incremental parsing algorithm that introduces exactly such a method for gradually changing the Markov order at the appropriate rate. This algorithm has been interpreted as a universal modeling scheme that sequentially calculates empirical probabilities in each context of the data, with the added advantage that the generated probabilities reflect contexts seen from the beginning of the parsed sequence to the current symbol. The LZ code length of any individual sequence attains the Markovian empirical entropy for any finite Markov order, i.e., the LZ algorithm in effect attains the Markov entropy of any given source, thereby functioning as a Universal Predictor.

LZ78 is a dictionary-based text compression algorithm that performs incremental parsing of an input sequence. This algorithm parses an input string "x_1 , x_2 , …. x_i" into $c(i)$ substrings "w_1 , w_2 , …. $w_{c(i)}$" such that for all j>0, the prefix of the substring w_j (i.e., all but the last character of w_j) is equal to some w_i for $1<i<j$. Because of this prefix property, parsed substrings can efficiently be maintained in a trie. Since LZ78 is a compression algorithm, it traditionally consists of two parts: the encoder and the decoder. Both encoder and decoder maintain dictionaries of phrases seen so far to encode/decode. In our case, however, we do not need to reconstruct the parsed sequence and therefore do not need to consider this as an encoder/decoder system, but simply a system that breaks up a given sequence (string) of states into phrases. From this perspective, Fig. 1 shows the pseudo code representation of LZ78 parsing.

Consider the sequence of input symbols x^n = "*aaababbbbbaabccddcbaaaa*". An LZ78 parsing of this string of input symbols would yield the following set of phrases: "*a,aa,b,ab,bb,bba,abc,c,d,dc,ba,aaa*". As described above, this algorithm maintains statistics for all contexts seen within the phrases w_i **.** For example, the context '*a*' occurs 5 times (at the beginning of the phrases "*a, aa, ab, abc, aaa*"), the context "*bb*" is seen 2 times ("*bb,bba*"), etc. These context statistics are stored in a trie. (Fig. 2).

```
initialize dictionary := null
initialize phrase w := null
loop
    wait for next symbol v
    if ((w.v) in dictionary):
        w := w.v
    else
        add (w.v) to dictionary
        w := null
        increment frequency for every
                possible prefix of phrase
    endif
forever
```

Fig. 1. The LZ78 algorithm.

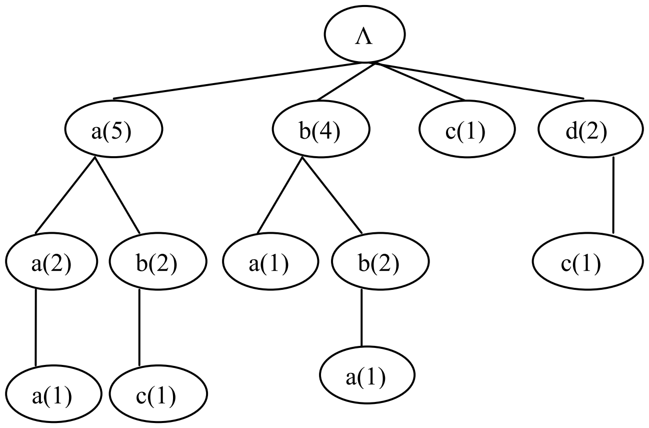

Fig. 2. Trie formed by LZ78 parsing.

4. Active LeZi

As shown in the previous section, a prediction scheme based on the LZ78 parsing algorithm can be viewed as a Universal Markov encoder with time-varying order, and therefore attains the desired FS predictability. In this section we describe the *Active LeZi* algorithm, which is constructed based on LZ78 compression.

4.1 *Problems in LZ78 parsing*

Any practical implementation of LZ78 suffers from the following two drawbacks:

i. In any LZ parsing of an input string, all the information crossing phrase boundaries is lost. In many situations, there will be significant patterns crossing phrase boundaries, and these patterns will affect the next symbol in the sequence.
ii. The convergence rate of LZ78 to the optimal predictability as defined above is slow. The results outlined in (Ref. 1) by Feder, et al state that LZ78 *asymptotically* approaches optimal predictability.

The authors have pointed out in a later commentary on the work (Ref. 6), that any practical implementation will have to address this issue.

Bhattacharya, et al. have addressed the issue of slow convergence rate to some extent in the *LeZi Update* algorithm (Ref. 4) , by keeping track of all possible contexts within a given phrase, and not just the prefixes to be found within a phrase. This method, however, does not address the issue of information lost across phrase boundaries.

4.2 *Active LeZi*

Active LeZi (ALZ) is an enhancement of LZ78 and *LeZi Update* (Ref. 4) that incorporates a sliding window approach to address the drawbacks outlined earlier. This approach also demonstrates various other desirable characteristics.

As the number of states seen in an input sequence grows, we can see that the amount of information being lost across the phrase boundaries increases rapidly. Our solution to this problem involves maintaining a variable length window of previously-seen symbols. We choose the length of the window at each stage to be equal to the length of the longest phrase seen in a classical LZ78 parsing. The reason for selecting this window size is that the LZ78 algorithm is essentially constructing an (approximation to an) order-*k-1* Markov model, where *k* is equal to the length of the longest LZ78 phrase seen so far. (See Fig. 2 and Fig. 4).

Within this window, we can now gather statistics on all possible contexts. This builds a better approximation to the order-k Markov model, because it has captured information about contexts in the input sequence that cross phrase boundaries in the classical LZ78 parsing. Therefore, we gain a better convergence rate to optimal predictability as well as greater predictive accuracy.
Fig. 3 illustrates the algorithm itself, and Fig. 4 shows the trie formed by the Active LeZi parsing of the input sequence "*aaababbbbbaabccddcbaaaa*".

We can see that this is a "more complete" order-*Max_LZ_length-1* Markov model than the one shown in Figure 2, in that it now incorporates more information about the contexts seen. (For our input sequence, the result is an order-2 Markov model).

Active LeZi demonstrates the following characteristics:

- It is in essence a growing-order Markov model that attains optimal FS predictability, due to the optimality of LZ78.
- As the length of the longest LZ78 phrase grows, ALZ stores more and more information, which implies that as the input sequence (the experience) grows, the algorithm performs better. This is a desirable characteristic of any learning algorithm.

The convergence to the optimal FS predictor is faster since ALZ now gathers information that was formerly inaccessible to the LZ78 parser.

```
initialize dictionary := null
initialize phrase w := null
initialize window := null
initialize Max_LZ_length = 0
loop
        wait for next symbol v
        if ((w.v) in dictionary):
                w := w.v
        else
                add (w.v) to dictionary
                update Max_LZ_length if necessary
                w := null
        endif

        add v to window
        if (length(window) > Max_LZ_length)
                delete window[0]
        endif
        Update frequencies of all possible
                contexts within window that includes v
forever
```

Fig. 3. *Active LeZi* algorithm.

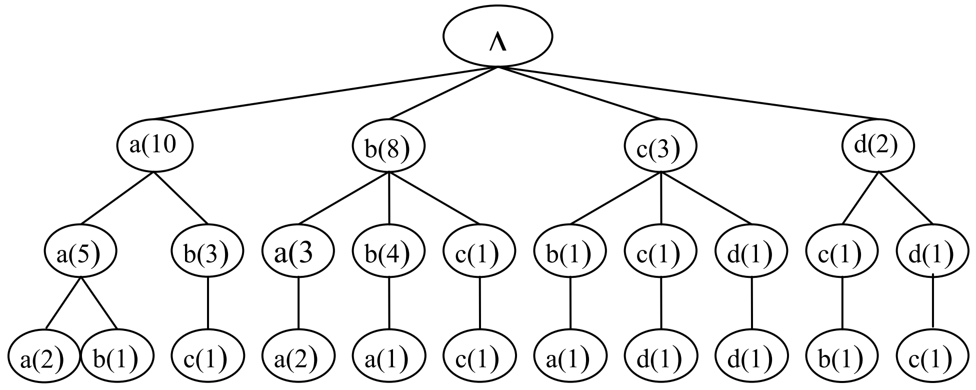

Fig. 4. Trie formed by the ALZ parsing of the string "*aaababbbbbaabccddcbaaaa*".

4.3 ***Probability assignments for prediction***

In order to predict the next event of the sequence for which ALZ has built a model, we calculate the probability of each state occurring in the sequence, and predict the one with the highest probability as the most likely next action.

It has been pointed out in (Ref. 6) that in order to achieve better convergence rates to optimal predictability, the predictor must "lock on" to the minimum possible set of states that is representative of the sequence being considered. For sequential prediction, it has been shown that this is possible by using a "mixture" of all possible order models in assigning the next symbol its probability estimate. For a method that considers different orders of models, we turn once again to data compression and the Prediction by Partial Match (PPM) family of predictors. This has been used to great effect in (Ref. 4), for a predictive framework based on LZ78.

PPM algorithms consider different-order Markov models in order to build a probability distribution by weighting different-order models appropriately. In our predictive scenario, Active LeZi builds an order-*k* Markov model. We now employ the PPM strategy of *exclusion* (Ref. 7) to gather information from models of order 1 through *k* to assign the next symbol its probability value. This method is illustrated by considering the example sequence used in the previous sections: "*aaababbbbbaabccddcbaaaa*".

The window maintained by Active LeZi represents the set of *contexts* used to compute the probability of the next symbol. In our example, the last phrase "*aaa*" (which is also the current ALZ window) is used. Within this phrase, the contexts that can be used are all suffixes within the phrase, except the window itself (i.e. "*aa*", "*a*", and the null context).

Suppose the probability that the next symbol is an *a* is being computed. From Fig 4 we see that an *a* occurs two out of the five times that the context "*aa*" appears, the other cases producing two null outcomes and one "*b*". Therefore the probability of encountering an "*a*" at the context "*aa*" is 2/5, and we now fall back (escape) to the order-1 context (i.e. the next lower order model) with probability 2/5. At the order-1 context, we see an "*a*" five out of the ten times that we see the "*a*" context, and of the remaining cases, we see two null outcomes. Therefore we predict the "*a*" at the order-1 context with probability 5/10, and escape to the order-0 model with probability 2/10. At the order 0 model, we see the "*a*" ten out of 23 symbols seen so far, and we therefore predict "*a*" with probability 10/23 at the null context. The *blended* probability of seeing an "*a*" as the next symbol is therefore:

$$\frac{2}{5}+\frac{2}{5}\left\{\frac{5}{10}+\frac{2}{10}\left(\frac{10}{23}\right)\right\}$$

Similarly, let us compute the probability that the next symbol is a "*c*". In this case, the order-2 and the order-1 contexts do not yield a "*c*". Therefore, we escape to the order-0 model and predict a "*c*" with a probability of 3/23. In this case the total probability of seeing a "c" would be

$$\frac{0}{5}+\frac{2}{5}\left\{\frac{0}{10}+\frac{2}{10}\left(\frac{3}{23}\right)\right\}=\frac{2}{5}*\frac{2}{10}*\frac{3}{23}.$$

This method of assigning probabilities has the following advantages:

- It solves the zero-frequency problem. In the above example, if only the longest context had been chosen to make a decision on probability, it would have returned a zero probability for the symbol "*c*", whereas lower-order models show that this probability is indeed non-zero.
- This blending strategy assigns greater weight to higher-order models in calculating probability if the symbol being considered is found in that context, while lower-order models are suppressed owing to the null context escape probability. This is in keeping with the advisability of making the *most informed* decision.

5. Application to a Smart Home Environment

The smart home provides a ready environment for employing sequential prediction algorithms such as ALZ. The MavHome smart home project at the University of Texas at Arlington characterizes the smart home as and intelligent agent whose goals include maximizing inhabitant comfort and optimizing energy usage (Ref. 8), by reducing interaction between the inhabitant and the home. To achieve this end, one of the tasks that the home has to perform is predict which of the devices in the home the inhabitant will interact with next, so that the activity may be automated. The home will have to make this prediction based only on previously-seen inhabitant interactions with various devices. From the comfort standpoint as well as for optimizing energy consumption, it is essential that the number of prediction errors that the house makes is kept to a minimum – not only would it be annoying for the inhabitant to have to reverse home decisions, but prediction errors will lead to energy wastage.

A smart home inhabitant typically interacts with various devices as part of his routine activities, interactions that may be considered as a sequence of events with some inherent pattern of recurrence. For example, our routines when we wake up in the morning are most likely the same everyday – turn on the kitchen light, turn on the coffee maker, turn on the bathroom light, turn off the bathroom light, turn on the toaster, turn off the coffee maker, etc.

Typically, each inhabitant-home interaction event '*e*', is characterized as a triple consisting of the device with which the user interacted, the change that occurred in that device, and the time of interaction. In this model we assume that devices are associated with the binary-valued ON and OFF states.

$$e = \langle Device\#,\ ON/OFF,\ TIME \rangle$$

The following tests were designed to evaluate the sequential prediction capability of ALZ, so the time information of the events was not considered. Each unique input symbol, x_t is therefore identified by the two-tuple consisting of the device ID, and the state change of that device.

ALZ has been tested on data obtained from a Synthetic Data Generator (SDG), which approximates the data that would likely be obtained in a real Home scenario. The SDG can be used to generate data from various configurable scenarios of user interaction.

The first set of tests was performed on data sets with a great deal of inherent repetitiveness, and lacking noise. The ALZ Learning Curves were plotted by testing the number of correct predictions from the next 100 events, while increasing the training data set. This generated the learning curve in Figure 5, and as can be seen the performance converges to 100% accuracy rapidly. This indicates that ALZ is a strong sequential predictor.

The second series of tests used SDG data sets of 2000 points generated from a set of 6 typical home scenarios incorporating 8-15 devices reflecting regular daily patterns of activity. These data sets were also seeded with randomly inserted events from 5-15% of the size of the data set. When averaged over the six scenarios, this setup yielded the learning curve shown in Figure 6, which converges to about 86% accuracy.

In addition to the tests performed on the simulated data, ALZ was also tested on data collected in an actual Smart Home environment – the AI Lab at the University of Texas at Arlington. This data was collected from monitoring the interactions of six different participants with devices, based on regular daily and weekly patterns of activity. These represent data that will typically be seen in a home environment – with all the associated randomness and noise within the system. As we can see from Figure 7, the algorithm converges to about 47% accuracy on one month of data consisting of about 750 data points.

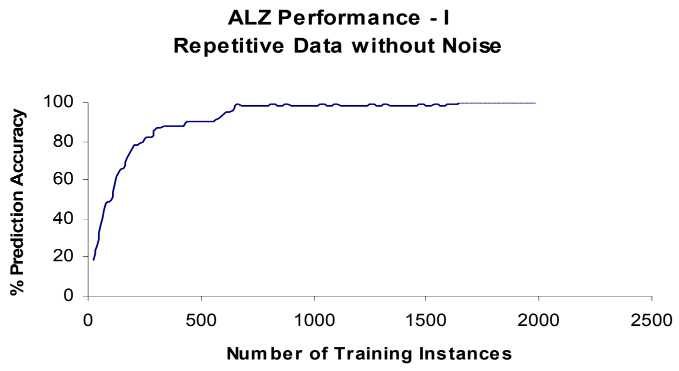

Fig. 5. ALZlearning curve on repetitive data without noise.

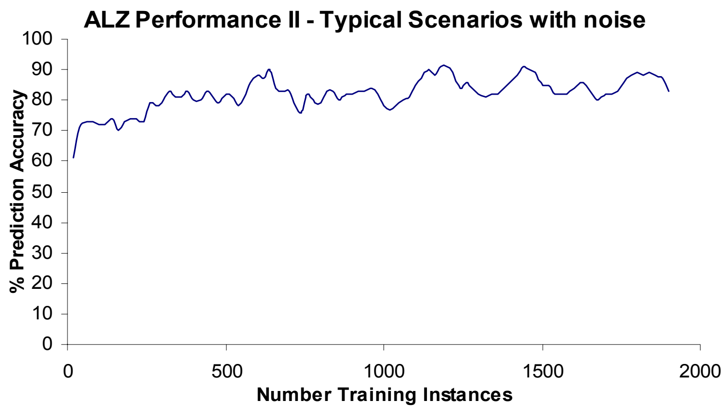

Fig. 6. ALZ learning curve on data with noise

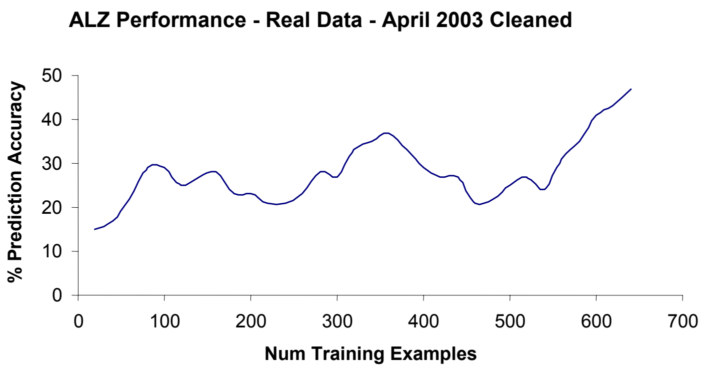

Fig. 7. Learning curve on actual smart home data collected in the AI Lab, UTA.

5.1 *Learning a Gaussian measure of relative time between events using ALZ*

In an agent-based smart home architecture, such as found in the MavHome, (Ref. 8, 9) the responses of the home to various situations are decided by the decision-making agent, which is aided in making the appropriate decision by various prediction agents, such as ALZ. The decision-making agent is greatly benefited if the prediction agent is capable of predicting what the next action is, as well as when it is going to occur, in order that the action may be more effectively automated.

In various typical home situations, the time between one inhabitant-home interaction and the next might depend on the particular sequence, or history of previous interactions. Given the assumption that such a dependency does indeed exist in certain sequential processes of events, a sequential predictor such as ALZ can be used to learn a measure of relative time interval between the various events.

In this model, the decision-making agent assumes that the next event to occur is already known, and given that information, requires an estimate of the time interval that is going to elapse before that event takes place. Assuming that the inhabitant-device interactions within the home are fairly regular, it is reasonable to suppose that the time interval between events approximate a normal distribution. Therefore, the decision-making agent requires the predictor to learn a Gaussian that represents the normal distribution of the time intervals between events, where each Gaussian is characterized by the mean 'μ', and the standard deviation 'σ'. The Gaussian of the time intervals is thus defined as:

$$G(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{(t-\mu)^2}{2\sigma^2}\right)} \tag{3}$$

5.1.1 Incrementally building a Gaussian measure of relative time for each phrase

In this model, the input to the predictor includes the time between the occurrence of this event and the previous event as well, i.e. each input event is defined by the 3-tuple *e* = < *Device ID, ON/OFF, Time*>.

Here, the relative frequency counts of the various phrases are stored as before in the trie, and in addition, each node in the trie also incrementally builds a Gaussian that represents the observed normal distribution of the relative time of occurrence of the last event in that phrase. In order to efficiently manage storage of the timing information, it is also essential that these Gaussians be constructed incrementally without having to store all of the observed timing information in the nodes of the trie. Of the two quantities that characterize the Gaussian, the mean μ is easily constructed incrementally. It can also be shown that the std. deviation σ can be recursively defined for *n* data points in terms of the mean and std. deviation for the previous *n-1* data points as:

$$\sigma_n = \left(\frac{1}{n}\right) \left\{(n-1)\left[\sigma_{n-1}^2 + (\mu_n - \mu_{n-1})^2\right] + (t_n - \mu_n)^2\right\} \tag{4}$$

Here, t_n represents the new data point. This recursive definition implies that the Gaussian can be built incrementally. Therefore, in this model, each node in the trie built by ALZ contains the Gaussian information about the relative time of occurrence of that event in the context of the previous events represented by the nodes along the path traced in the trie in getting to that node.

5.1.2 Building a composite Gaussian representing the relative time during prediction

Given a predicted event, the time of occurrence of that event is to be computed using the model of time built as discussed above. As with the prediction of the event itself, described in Sect. 4.3, information stored at various orders of the trie must be blended to strengthen the resulting prediction. That is, the

Gaussians that have been built for the various phrases within the ALZ window must be merged after weighting them appropriately.

In order to merge various Gaussians, a set of data points is generated for each Gaussian, based on Equation 3 to create a pool of data points. The number of points that each Gaussian contributes to this pool depends on the weighting of that particular Gaussian. To determine the weights for each of the Gaussians, we once again turn to the PPM method described in Section 4.3, of assigning a higher weight to the higher-order models, and recursively assigning lower weights to the lower-order models. The following example illustrates this method and is based on the example in Section 4.3, and the trie in Figure 4.

For the trie and sequence in Fig. 4, the next symbol predicted by ALZ is an '*a*'. Therefore, the Gaussians are drawn from the phrases at orders 2 and 1 (i.e. the phrases '*aaa*' and '*aa*'). The weight of a particular order of the model is the same as the contribution of that order to the composite PPM probability as computed in Section 4.3. and is equal to the product of all the escape probabilities to get to that order and the context probability at that order. Therefore, the weight of the order-2 model in our example would be $1 * \frac{2}{5} = 0.4$, and the weight of the order-1 model in this example would be $\frac{2}{5} * \frac{5}{10} = 0.2$. The number of points generated with the Gaussian at the phrase 'aaa' would therefore be twice as large as that generated with the Gaussian at the phrase 'aa'.

Then the mean and standard deviation of this pool of points is computed and represents the measure of the time that is expected to elapse before the next predicted event occurs.

5.1.3 Performance of ALZ in learning Gaussians

The performance of the above method in learning a Gaussian measure of relative time was evaluated using a synthetically generated data set. The data consisted of a pattern of repeating events, representative of a sequential pattern typically encountered in the smart home scenario, where the order of events was constant, but the time difference between successive events were drawn from different distributions depending on the context in which that event was taking place. Therefore, once the algorithm converged to predicting the next event correctly, the method of learning time patterns could be evaluated effectively. As a measure of the performance, the actual time of occurrence of the next event was compared to the predicted distribution. These results are summarized in the following table.

Table 1. Results of learning a Gaussian measure of relative time.

AREA OF DISTRIBUTION	% CANDIDACY OF NEXT PREDICTED EVENT IN THIS AREA OF DISTRIBUTION
$\mu \pm 0.5\,\sigma$	45%
$\mu \pm \sigma$	70%
$\mu \pm 2\,\sigma$	92%

The above table reflects that for example, 70% of the time, the next event occurred within mean ± std. dev. of the predicted time, etc. This experiment validates that ALZ is indeed capable of learning a distribution of relative time that is dependent on the history of events preceding the current event. These results imply that ALZ can indeed effectively learn a Gaussian measure of time in a sequential process.

6. Conclusions

ALZ effectively models sequential processes, and is extremely useful for prediction of processes where events are dependent on the previous event history. This is because of the ability of the algorithm to build an accurate model of the source of the events being generated, a feature inherited from its information theoretic background and the LZ78 text compression algorithm. The effectiveness of the method for learning a measure of time can also be attributed to fact that ALZ is a strong sequential predictor. The sound theoretical principles on which ALZ is founded also mean that ALZ is an optimal Universal Predictor, and can be used in a variety of prediction scenarios.

Acknowledgements

This work was supported in part by National Science Foundation (NSF) grant IIS-0121297.

References

1. M. Feder, N. Merhav, and M. Gutman, Universal Prediction of Individual Sequences, in *IEEE Transactions on Information Theory*, Vol 38, No.4 (1992).
2. E. Federovsky, M. Feder, and S. Weiss, Branch Prediction based on Universal Data Compression Algorithms, in *Proceedings of the Annual. Symposium on Computer Architecture* (1998), pp. 62-72.
3. J.S. Vitter and P. Krishnan, Optimal Prefetching via data compression, in *Journal of the ACM*, Vol 43 No. 5 (1996), 771-793.
4. A. Bhattacharya and S.K. Das, LeZi-Update: An Information-theoretic framework for personal mobility tracking in PCS networks, in *ACM/Kluwer Wireless Networks Journal*, vol. 8, no. 2-3 (2002), 121-135.
5. J. Ziv and A. Lempel, Compression of Individual Sequences via Variable Rate Coding, in *IEEE Transactions on Information Theory*, vol. IT-24 (1978), 530-536.
6. M. Feder, N. Merhav, and M. Gutman, Reflections on 'Universal Prediction of Individual Sequences' , Invited article, in *IEEE Information Theory Society Newsletter*, (1994).
7. T.C. Bell, J.G. Cleary, I.H. Witten, *Text Compression*, (Prentice Hall Advanced Reference Series, 1990).
8. S. Das, D.J. Cook, A. Bhattacharya, E. Heierman, and T. Lin, The Role of Prediction Algorithms in the MavHome Smart Home Architecture, in *IEEE Personal Wireless Communications,* Vol 9 No. 6, (2002), 77-84.
9. D.J. Cook, M. Huber, K. Gopalratnam, and E. Heierman, Learning to Control a Smart Home Environment., to appear in *IEEE Transactions on Systems, Man & Cybernetics*.
10. T. Cover and J. Thomas J, *Elements of Information Theor*y (Wiley, 1991).