

Lazy Learning by Scanning Memory Image Lattice *

Yiqiu Han and Wai Lam

Department of Systems Engineering and Engineering Management

The Chinese University of Hong Kong

Shatin

Hong Kong

{yqhan,wlam}@se.cuhk.edu.hk

keywords: supervised learning, lazy learning, instance-based learning

Abstract

SMILE (Scanning Memory Image Lattice) is a lazy learning framework based on a memory image lattice scanning technique. To classify an unseen instance, the instances in the training set will generate a memory image lattice in terms of the similarities between the training instances and the unseen instance. An exploration algorithm of memory image lattice is designed to search an appropriate set of images of training instances to produce the final prediction. SMILE differs from other lazy learning algorithms in that it utilizes subsets of attribute values as much as possible. This design leads to a more flexible model which is less sensitive to data sparseness.

1 Introduction

We develop a lazy learning framework based on a memory image lattice that reflects the similarities between the training instances and the unseen instance. Lazy learning models [1] do not involve any model construction before they encounter the unseen instance, implying that they do not have any processing until they are requested to predict the class label of an unseen instance. The model and all the intermediate results are discarded when the learning process for this unseen instance ends. Some lazy learning algorithms are also described as “instance-based” or “memory-based” where “memory” mainly refers to training instances.

Lazy learning algorithms [10, 8, 4] differ from common eager algorithms, which eagerly compile the training data into some concept descriptions (e.g. rule sets, decision trees, networks, graphical model). In general,

lazy learning algorithms need much less training costs but more storage and computational resources than eager algorithms during operation or testing. Many eager learning algorithms have to work under a particular hypothesis, which covers the entire instance space, while lazy learning algorithms can make use of a richer hypothesis space by using the characteristics of the unseen instance to explore different hypotheses during operation. Thus in many learning tasks, lazy learning algorithms can be a powerful substitute. In fact some lazy methods such as local methods proposed by [3] outperform some eager algorithms.

The family of k-nearest neighbor (kNN) learning algorithms and its variants [7, 5, 6] is one of the most classical and widely adopted algorithms among lazy learning methods. Actually the idea of kNN is quite simple. It collects k nearest training instances in terms of similarities between their attribute values and those of the unseen instance, and then generates the final output by examining these nearest instances. However, in many situations, there may not be enough qualified nearest neighbors.

Unlike many existing lazy learning methods, our model considers the subsets of attribute values of the unseen instance rather than the Euclidean distances between the unseen instance and the training instances. These subsets compose a lattice and each joint represents a node in the lattice. If one training instance shares the same or similar values with the unseen instance on a joint, the joint will keep a copy of the training instance called *memory image*. In this paper, we refer the term “memory” to the stored information, i.e., training instances.

Our model uses the lattice to search an appropriate set of attribute subsets via an exploration process. Those attribute subsets are treated as memory pools to collect all the memory images inside. Rather than using the training instances directly to combine the final result, a memory image combination technique

*The work described in this paper was substantially supported by grants from the Research Grant Council of the Hong Kong Special Administrative Region, China (Project Nos: CUHK 4187/01E and CUHK 4179/03E) and CUHK Strategic Grant (No: 4410001).

is developed to produce the final prediction. These characteristics make our model more flexible and less sensitive to data sparseness than other instance-based lazy learning models. When there are enough training data and complete information for all instances, our model can perform in the same way as kNN does. When there are many missing values affecting the reliable calculation of the Euclidean distance as required by kNN, our model can still give a fairly satisfactory performance.

In essence, two strategies are specially designed to explore the lattice. The first strategy is to ensure the selected subsets are among the nearest ones to the summit of the lattice, that is, nearest to the unseen instance. The second strategy is to ensure that every selected joint contains enough memory images to produce reliable learning result no matter which kind of combining method is used. These strategies will guide the search on the distribution of memory images among the lattice for an appropriate set of subsets. All the memory images contained by those subsets are combined to make the final prediction.

2 Background of SMILE Framework

In the following discussions, we use capital letters such as C, D for variable names, and lower-case letters such as c, d for specific values of these variables. The size of the set X is denoted by $|X|$.

Consider a classifier learning problem, where an instance is associated with a set of attributes and a class label. Suppose F is the whole set of attributes consisting of m discrete-valued attribute variables from F_1 to F_m .¹ C , a discrete-valued variable, represents the class label. An instance d in the collection will be represented as $F(d) = (F_1(d), F_2(d), \dots, F_m(d), C(d) = c)$. $F_i(d)$ represents the value of attribute F_i for instance d . To obtain the membership likelihood of class C for a new instance t , a likelihood score $L(C|F(t))$ will be estimated based on the information represented by $F(t)$.

Definition An attribute joint A is defined as a subset of the full attribute set F . Specifically, $A \subseteq F$.

Definition Given an attribute joint $A \subseteq F$, we say two instances d and t are similar with respect to A , denoted by $\text{SIM}_A(d, t)$, if and only if for every $A_i \in A$; $A_i(d) = A_i(t)$.

Definition Given $A \subseteq F$ and a target instance t , a *memory pool* $S(A, t)$ can be obtained from the training

data by selecting those instances similar to t regarding A as follows: $S(A, t) = \{d \mid \text{SIM}_A(d, t)\}$

Definition We define $S(A, t, c)$ the memory images in $S(A, t)$ that have the same class label as C : $S(A, t, c) = \{d \mid \text{SIM}_A(d, t) \wedge C(d) = c\}$

Suppose we wish to predict the class C for a target unseen instance t . Let the set of attributes we used in learning be A where $A \subseteq F$. Many learning models treat A as the full set of attributes F , but models with feature selection consider $A \subseteq F$. One common technique for using the instantiation $A(t)$ to predict C of t is to compute a likelihood score $L(C|A(t))$ for each class label:

$$L(C|A(t)) = \hat{P}(C|A(t)) = \frac{\hat{P}(C, A(t))}{\hat{P}(A(t))} \quad (2.1)$$

A straightforward method is to make use of $S(A, t)$ and $S(A, t, C)$ to estimate $\hat{P}(A(t))$ and $\hat{P}(C, A(t))$ respectively. For the multi-classification learning task, this method is actually conducting majority voting in the memory pool $S(A, t)$. In practice, it is not easy to find a good attribute joint A to produce reliable estimations for classification. If the attribute joint consists of too many attributes, it will be difficult to collect enough memory images in $S(A, t)$ to produce reliable classification or prediction. kNN attempts to tackle this problem by choosing k nearest training instances rather than $S(F, t)$ to conduct majority voting. On the other hand, if the attribute joint A consists of too few attributes, it will lose some useful information. Naive Bayesian learning model attempts to solve this problem by introducing the conditional independence assumption among all attributes given a class label.

We develop a framework called SMILE (Scanning Memory Image Lattice) that uses multiple attribute subsets. Our framework first searches for a proper set of attribute subsets, denoted by U , which is a subset of the power set \mathfrak{S}_F (i.e., $U \subseteq \mathfrak{S}_F$). The technique of finding U is to avoid the problems occurred when we use just one attribute joint as mentioned earlier. To achieve this goal, we consider the memory image lattice. An exploration algorithm is developed to find out an appropriate set of attribute subsets by exploring the memory image lattice.

After a suitable U is obtained, for each attribute joint $U_i \in U$, the corresponding *memory pool* $S(U_i, t)$ will be constructed. Classifications will be conducted by combining all the memory images in those memory pools to produce the final learning result, denoted by $L(C|F(t))$.

¹We only consider discrete values in this paper and all continuous values are discretized before processing. The similarity is simply defined as the equality of discrete values.

3 Memory Image Lattice

3.1 Construction of the Lattice Suppose there are m attributes, i.e., $|F| = m$. Then we have $|\mathfrak{S}_F| = 2^m$. The number of possible subsets of \mathfrak{S}_F is 2^{2^m} . Obviously it is extremely difficult to examine all possible sets of attribute subsets. To handle this problem, all attribute subsets in \mathfrak{S}_F will be organized into a partially-ordered lattice. Fig. 1 illustrates such a lattice for a problem where $|F| = 4$.

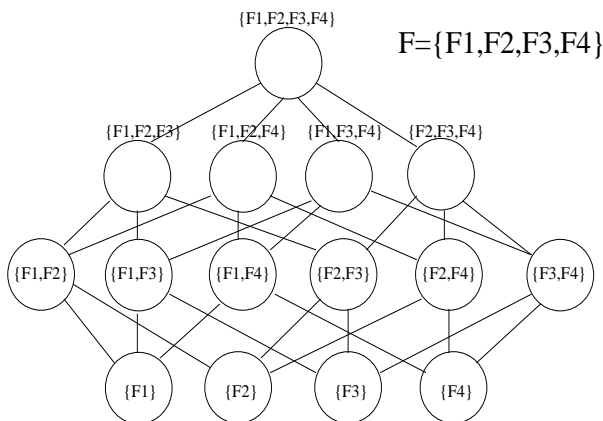


Figure 1: Lattice composed of attribute subsets

A node in the lattice represents an attribute joint. Every node A on the lattice will maintain a corresponding *memory pool* $S(A, t)$. One training instance may have multiple copies, known as memory images, in different memory pools. This memory image lattice can be intuitively understood as the human mind that is trying to perceive a new object. While bearing some characteristics of the object, similar memory images, which share similar characteristics with the object, will appear in the mind to help the perceiving process. If we change the perspective of observing the object, different sets of characteristics will produce different memory images. Human often use a set of low dimensional perspectives to help understand a high dimensional object. Our model is inspired by this process. It selects a set of attribute subsets to help collect memory images for predicting the class label.

The link between two nodes denotes the subset relationship between different attribute subsets. It can be viewed as a kind of partial order relationship called *cover*.

Definition For any nonempty element A and B in \mathfrak{S}_F , A covers B if $B \subseteq A$.

If A covers B , then the following expressions hold:

$$\text{SIM}_A(d, t) \Rightarrow \text{SIM}_B(d, t); \quad (3.2)$$

$$S(A, t) \subseteq S(B, t) \quad (3.3)$$

3.2 Strategies for Exploring the Lattice On average, given A covers B , the memory images in $S(A, t)$ are believed to be more similar to t than those in $S(B, t)$ because memory images in $S(A, t)$ may share more identical attributes with t . So the memory images from $S(A, t)$ are more valuable in learning. In the learning process, if $S(A, t)$ is used and $|S(A, t)|$ is large enough, then $S(B, t)$ can be ignored.

The first strategy is to explore the large lattice efficiently, and utilize the information underlying the attribute value subsets as much as possible. It scans the whole lattice in a descending top-down sequence as described by the following steps:

1. Try to add to U the attribute subsets containing as many attributes as possible.
2. Once an attribute joint in the memory image lattice is selected and added to U , all the attribute subsets covered by that joint can be trimmed from the lattice to avoid being selected into U .

This strategy can be intuitively understood as simulating the thinking process of the human. When one person needs to make a decision under a particular circumstance, one will attempt to recall some similar scenarios. If there are many scenarios that are very similar with the current situation, other less similar scenarios will be suppressed in his mind. This strategy can accelerate the processing of memory image lattice. Hence any two elements of U should not cover each other as described in the following property:

$$\forall (U_i \in U, A \in \mathfrak{S}_F; A \neq U_i \wedge ((U_i \subset A) \cup (A \subset U_i))) \Rightarrow A \notin U \quad (3.4)$$

The second strategy for exploring the lattice is to avoid the root node or other nodes close to the top of the lattice being improperly selected in view of data sparseness. It imposes a threshold α on the number of memory images for any node that is going to be added to U . Otherwise this model will degenerate to Eqn. 2.1 with $A = F$, and suffer from the problem of data sparseness. Therefore the following constraint is imposed on U :

$$U = \{U_i | U_i \in \mathfrak{S}_F \wedge |S(U_i, t)| > \alpha\} \quad (3.5)$$

U should make use of the information offered by F as much as possible. So U should cover every $F_i \in F$:

$$\forall F_i \in F, \exists U_i \in U \text{ so that } F_i \in U_i$$

$$\forall A \subseteq F, |S(A, t)| > \alpha \Rightarrow \exists U_i \in U, A \subseteq U_i \quad (3.6)$$

Note that, in general, the final set U of attribute subsets can be overlapping or non-overlapping. The non-overlapping case forms a partitioning of attributes.

As mentioned earlier, one characteristic about memory image collection for U is that it will keep multiple copies of a training instance. Different A_i may have the same copy of a training instance. If $A \notin U$ and $\exists U_i \in U, U_i \subset A$, all memory images of A should still be found in the memory image collection of U . The reason is that all U_i covered by A will have a copy of these more useful memory images. The number of copies of a training instance appearing in the memory image collection of U becomes a weight which measures the similarity between that training instance and t .

The time complexity of SMILE has an upper bound of $O(x \log x)$ where x is the number of training instances having at least one identical attribute value with the unseen instance to be classified.

3.3 Lattice Exploration Process In order to obtain a suitable U satisfying the properties discussed above, we develop a top-down breadth-first search process to explore the memory image lattice. The search is guided by pruning based on the property of cover and the size of memory pools. The discussions on U in the previous subsection already provide a strategy for pruning the nodes, which remain to be explored in the memory image lattice. However, the storage and computational cost are still very high even for moderate number of attributes. To solve this problem, we make use of memory pools in the search process. We define a concept *maximal overlap attribute joint*.

Definition Let d be a training instance and t be an instance to be classified. The maximal overlap attribute joint $M(d, t)$ is the maximal attribute subset on which d and t agree.

We maintain an *active memory pool* E which initially contains all the training instances satisfying $|M(d, t)| > 0$. In the search process, this active memory pool is dynamically maintained to support both the pruning in memory image lattice and the construction of U . The whole memory image lattice structure need not be explicitly constructed during the search process. When the search is conducted at the i th level of the memory image lattice in top-down manner, we only need to consider the nodes as follows:

$$\{M(d, t) | d \in E \wedge |M(d, t)| = i\} \quad (3.7)$$

Attribute joint A will be recorded and added into U only when $|S(A, t)| > \alpha$. After A is added to U , all the attribute subsets covered by A are eliminated

by removing the following instances from the active memory pool E :

$$\{d | M(d, t) \subseteq A \wedge d \in E\} \quad (3.8)$$

This process continues until all the valid attribute subsets have been removed or selected. In other words, no instances remain in E .

4 Memory Image Combination

In the Naive Bayesian learning model, the likelihood score is calculated by the multiplication of the probabilistic parameters of all individual attributes F_i . In our SMILE framework, we do not assume conditional independence among different attributes and propose another combination technique to trade the precision for robustness. Rather than intersection, we consider the union of all attribute sets U_i , i.e., $U(t) = \bigcup U_i(t)$. We use $|S(U_i, t)|$ to estimate $P(U_i(t))$, and $|S(U_i, t, C)|$ to estimate $P(C, U_i(t))$.

According to the probability theory, for any n events, suppose E_j^k is the j th set composed of k events. Then we have:

$$P(\cup_{i=1}^n E_i) = \sum_{k=1}^n (-1)^{k-1} \sum_j E_j^k \quad (4.9)$$

Apply Eqn. 4.9 to U ($n = |U|$):

$$P(\cup_{i=1}^n U_i(t)) = \sum_{k=1}^n (-1)^{k-1} \sum_j (\wedge_{i=1}^k U_{j_i}^k(t)) \quad (4.10)$$

According to the properties of U , we have:

$$\forall U_i, U_j \in U; U_i \cup U_j \notin U \quad (4.11)$$

For any $(\wedge U_i(t))$, $P(\wedge U_i(t)) = P(A(t))$ where $A = (\cup U_i)$. Eqn. 4.11 shows that $A \notin U$ and $S(A, t)$ must be smaller than α otherwise A should be selected rather than all of $U_i \subseteq A$. Hence only the first order items in Eqn. 4.10 need to be considered:

$$P(U(t)) = P(\cup_i(U_i(t))) \simeq \sum_i P(U_i(t)) \quad (4.12)$$

Similarly we have:

$$P(C, U(t)) = P(C, \cup_i(U_i(t))) \simeq \sum_i P(C, U_i(t)) \quad (4.13)$$

The likelihood score computed by the union of attribute sets is more robust than that of the intersection of attribute sets. We make use of $|S(U_i, t)|$ and

$|S(U_i, t, C)|$ to estimate $\hat{P}(U_i(t))$ and $\hat{P}(C, U_i(t))$ respectively. As a result:

$$\begin{aligned}
 L(C|F(t)) &= \frac{\sum_{i=1}^m \hat{P}(C, U_i(t))}{\sum_{i=1}^m \hat{P}(U_i(t))} \\
 &= \frac{\sum_{i=1}^{|U|} |S(U_i, t, C)|}{\sum_{i=1}^{|U|} |S(U_i, t)|} \\
 &= \frac{\sum_{i=1}^{|U|} (|S(U_i, t)|L(C|U_i(t)))}{\sum_{i=1}^{|U|} |S(U_i, t)|} \quad (4.14)
 \end{aligned}$$

```

1 For every instance  $t$  to be classified
2   Initialize  $U$  to empty
   //Build an active memory pool  $E$ 
3 For every training instance  $d_i$ 
4   If  $|M(d_i, t)| > 0$ 
5     Add  $d_i$  to  $E$ 
6      $EMAX = \text{MAX}(|M(d_i, t)|)$ 
   //Search process
7 For  $j = EMAX$  to 1 do
8   For every  $|M(d_i, t)| = j, \forall d_i \in E$ 
9     If  $|S(M(d_i, t), t)| > \alpha$ 
10      Add  $M(d_i, t)$  to  $U$ 
      //Prune nodes
11     For every  $d_k \in E$ 
12       If  $M(d_k, t) \subseteq M(d_i, t)$ 
13         Remove  $d_k$  from  $E$ 

```

Figure 2: Exploration process of the memory image lattice

In fact $(\sum_{i=1}^{|U|} (|S(U_i, t)|L(C|U_i(t))))/(\sum_{i=1}^{|U|} |S(U_i, t)|)$ is a weighted summation. The weight is the size of $S(U_i, t)$. The set with more instances will be assigned larger weight because its parameter estimation is more reliable.

5 Preliminary Experiments

We have conducted preliminary experiments on 5 benchmark data sets from the UCI repository of machine learning database [2]. They are either classical or with extreme number of attributes or instances. We partitioned each data set into 10 even portions and then conducted 10-fold cross-validation. The mean and the standard deviation of the accuracies of 10-fold cross-validation are used to measure the performance. For comparison, we also investigate the performance of Naive Bayesian, SVM, kNN, and J48, which are from the Weka-3-2-2 machine learning software [9]. The results indicate that SMILE has superior or at least comparable performance on most of the data sets in comparison with other classical learning models. For data sets with large size and large attribute set, SMILE work

efficiently due to the efficient lattice exploration design. SMILE also shows benefits on small data sets in terms of accuracy. The intelligent search process and the introduction of memory images enable SMILE to handle data sparseness effectively.

Data Set	SMILE	kNN	Naive Bayesian	J48	SVM
Iris	94.0	96.0	96.0	95.3	85.3
Letter	86.8	94.8	64.2	87.8	81.7
Sonar	84.6	73.0	65.9	74.1	77.8
Weather	80.0	70.0	70.0	65.0	40.0
Zoo	98.1	88.2	95.2	92.1	92.1

Table 1: Classification performance of SMILE and other classifiers. The performance is measured by classification accuracy (in percentage) and standard deviation of 10-fold cross validation.

References

- [1] D. W. Aha. Editorial. *Artificial Intelligence Review, Special Issue on Lazy Learning*, 11:7–10, February 1997.
- [2] C. Blake, E. Keogh, and C. Merz. UCI repository of machine learning databases. <http://www.ics.uci.edu/~mlearn/MLRepository.html>.
- [3] L. Bottou and V. Vapnik. Local learning algorithms. *Neural Computation*, 4:888–900, 1992.
- [4] W. Daelemans, A. Bosch, and T. Weijters. IGTree: Using trees for compression and classification in lazy learning algorithms. *Artificial Intelligence Review, Special Issue on Lazy Learning*, 11:407–423, February 1997.
- [5] B. V. Dasarathy. *Nearest neighbor (NN) norms: NN pattern classification techniques*. IEEE Computer Society Press, 1991.
- [6] B. V. Dasarathy. Minimal consistent set (MCS) identification for optimal nearest neighbor decision systems design. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(3):511–517, MARCH 1994.
- [7] J. Friedman. Flexible metric nearest neighbor classification. Technical report, Stanford University, November 1994.
- [8] O. Maron and A. W. Moore. The racing algorithm: Model selection for lazy learners. *Artificial Intelligence Review, Special Issue on Lazy Learning*, 11:193–225, February 1997.
- [9] I. Witten and E. Frank. *Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.
- [10] Z. Zijian, G. Webb, and K. Ting. Lazy bayesian rules: A lazy semi-naive Bayesian learning technique competitive to boosting decision trees. In *Proceedings of the Sixteenth International Conference on Machine Learning (ICML)*, pages 493–503, 1999.