

# A novel content classification scheme for web caches

G. P. Sajeev · M. P. Sebastian

Received: 8 August 2010 / Accepted: 2 December 2010 / Published online: 28 December 2010  
© Springer-Verlag 2010

**Abstract** Web caches are useful in reducing the user perceived latencies and web traffic congestion. Multi-level classification of web objects in caching is relatively an unexplored area. This paper proposes a novel classification scheme for web cache objects which utilizes a multinomial logistic regression (MLR) technique. The MLR model is trained to classify web objects using the information extracted from web logs. We introduce a novel grading parameter *worthiness* as a key for the object classification. Simulations are carried out with the datasets generated from real world trace files using the classifier in Least Recently Used-Class Based (LRU-C) and Least Recently Used-Multilevel Classes (LRU-M) cache models. Test results confirm that the proposed model has good online learning and prediction capability and suggest that the proposed approach is applicable to adaptive caching.

**Keywords** Web caching · Predictive caching · Adaptive caching · Content classification · Performance · Logistic regression

## 1 Introduction

Caching and replication of web contents are well accepted techniques to improve the performance of web services. A

proxy-cache deployed near to a client serves the web documents locally as shown in Fig. 1. This increases the client side download speed and reduces the outborn traffic. An efficient proxy-cache server can deliver most of the requests from its local cache. The performance of such a service depends on the cache architecture, admission policy, replacement method and cache consistency. Handling the input traffic dynamics is the key aspect in a cache server's success.

With the introduction of web cache systems, traffic characteristics like self-similarity (Dill et al. 2002), popularity of web objects (Breslau et al. 1999) and web sites (Krashakov et al. 2006) were identified as important. These characteristics are utilized to optimize and improve the web cache performance using better cache replacement algorithms, cache architecture and cache consistency. The web is highly dynamic in nature with respect to the type of applications, growth, number of users and services. Hence, the reported traffic properties and their values are liable to change continuously. Thus, generalizing the characteristics of the web traffic is considered to be difficult (Cobb and ElAarag 2008).

The web caching task includes object caching, object replacement and consistency handling of an object. Each of the subtask arises in the following situations:

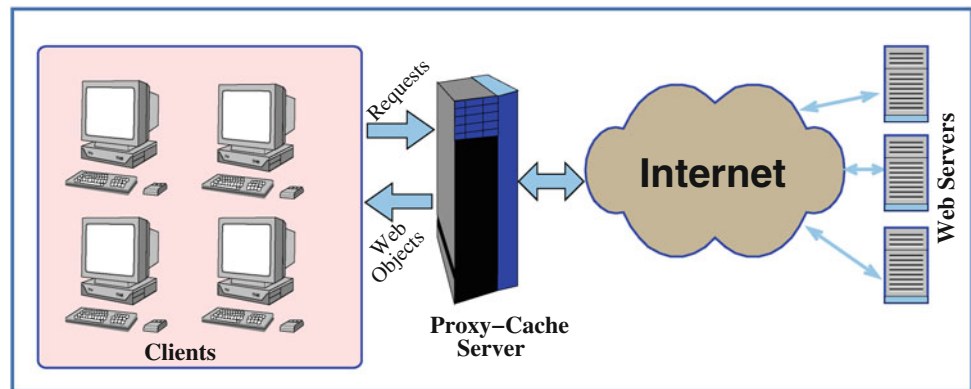
1. *Caching* to decide whether an object is to be cached or admitted to the cache store, when it arrives from the origin server.
2. *Replacement* to decide whether an object is to be replaced/evicted from the cache store, when the cache storage is full. This is the process of finding out the best candidate for eviction.
3. *Consistency* to decide an action (update/pre-fetch), when an object in the cache store becomes stale.

---

G. P. Sajeev (✉)  
Government Engineering College Kozhikode,  
WestHill, Kozhikode 673005, Kerala, India  
e-mail: gpsajeev@geckkd.ac.in

M. P. Sebastian  
Indian Institute of Management Kozhikode (IIMK),  
Calicut 673570, Kerala, India  
e-mail: sebasmp@iimk.ac.in

**Fig. 1** A proxy-cache server system



The above mentioned tasks are simple object selection issues, in which an object is selected using some grading value. The basic caching policies estimate this grading value using certain object request statistics like recency and frequency. The corresponding algorithms are known as LRU (Least Recently Used) and LFU (Least Frequently Used), respectively. These schemes are simple to implement and perform well in the case of uniform objects and latencies. As they cannot adapt with the size of the object and changes in traffic patterns, they are not good enough for the web caching. Extended methods like Greedy Dual-Size Popularity (GDSP) (Jin and Bestavros 2000), LUV (Least Unified Value) (Bahn et al. 2002) and Grade (Bian and Chen 2008) use a combination of two or more object attributes for decision making. They generally perform better than the basic caching schemes. The adaptive and intelligent caching schemes estimate the grading value by using the request statistics, network information and web server's capacity, which utilizes statistical or machine learning techniques (Sulaiman et al. 2008; Cobb and ElAarag 2008; Yang and Zhang 2003). The grading value used in these methods captures the traffic dynamics better, but with more computational overhead.

Podlipnig et al. (Podlipnig and Böszörmenyi 2003) conduct an extensive survey on cache replacement schemes. They claim that the caching schemes are influenced by four factors (characteristics) while applying a caching or a replacement strategy. The factors are frequency, recency, size and cost. However, it is very difficult to keep track of the history of all these factors simultaneously. Hence most of the existing approaches rely upon some object grading mechanism, like *popularity-rank* of the object (Chen and Zhang 2003; Jin and Bestavros 2000), *cost function* for retrieval of an object (Li et al. 2007; Cao and Irani 2002) and *page grade* (Bian and Chen 2008). All these approaches assume some general characteristics to define the grading parameter. These methods are not applicable if the traffic nature is dynamic. To tackle this situation, researchers adopted intelligent and heavy

machine learning techniques in the form of neural networks (NN) (Cobb and ElAarag 2008), artificial intelligence (AI) (Sulaiman et al. 2008) and web log mining technique (Yang and Zhang 2003) for which the computational overheads are high. Also, the practical feasibility of such methods is yet to be established. So, we propose a semi intelligent classification method (for object grading) which uses a light-weight machine learning technique and is adaptable to the dynamic nature of the web traffic.

The beneficiaries of caching are both users and system resources. Users will experience an immediate delivery of the object, as it is delivered from a local cache server. The demand for network resources such as bandwidth and server capacity will be reduced when the objects are cached. Generally, these benefits are represented in the form of two performance metrics named as hit ratio (HR) and byte hit ratio (BHR), which are borrowed from the conventional processor cache. Intuitively, HR indicates the benefit of caching from the user's perspective and the BHR reflects the cache benefits for the network resources. Note that in a processor cache, all the object attributes (size, latency and type) are uniform. Though HR and BHR are indicative of the performance in terms of percentage of saving, they are only very peripheral when the complex nature of web caching is considered. So, there is a need for other meaningful metrics which combine the object attributes and the benefits of caching.

Caching schemes can be developed using an object classification approach. In fact, an inherent (or hidden) classification works in all caching schemes, except for the basic caching algorithms. For instance, the model described by (Jin and Bestavros 2000), classifies objects into two classes; cache-able and non cache-able, according to its popularity and size. A classification scheme of user requests for objects as cache-able and non cache-able is given in (Cobb and ElAarag 2008), which uses the neural network (NN) as the classifier. To the best of our knowledge, only binary classification schemes are available for web cache objects, which use cache-ability of the object as

the key for classification. Motivated by this fact, we propose a generic classification scheme, which classifies the web objects into multilevel classes with reasonable prediction capability. We believe that this classification method can perform better than the binary classifiers. Our approach is different from the earlier classification approaches by constructing a model for the object *worthiness* with the informations extracted from the web traffic. This approach incorporates traffic patterns and object properties and can be extended or modified for a specific form of caching; examples include multimedia caching, server side caching and cooperative caching.

Logistic regression (LR) models are widely used in a number of applications, particularly in the medical field for the classification of data. Other classifier tools of interest are Artificial Neural Network (ANN), Support Vector Machine (SVM), K-nearest Neighbors and Decision Trees. Among these, LR and ANN methods are quite popular (Dreiseitl et al. 2001). The reported results in (Long et al. 1993) suggest that there is no significant performance difference between LR and Decision Trees. We have not considered the K-nearest Neighbors as its classification performance is not good enough to be comparable with LR and ANN (Dreiseitl and Ohno-Machado 2002).

The LR models have many advantages over other machine learning models (Sargent 2001). They are simple and flexible compared to the other models (Dreiseitl et al. 2001; Sargent 2001). The interpretation of results, ease of use and implementation support in various software packages are the main attractions of the LR method. Based on the observations, we choose MLR as the classifier (which has less computational overhead and simplicity, compared to the NN counterpart). Also, our caching model does not demand very high classification accuracy which is offered by the neural network classifiers (Green et al. 2006). The major contributions of this work are:

- Proposing a novel content classification scheme, which classifies the web objects into multiple classes.
- Computing a new comprehensive quality parameter *worthiness* for web objects, to help in caching decisions.
- Providing an experimentation showing that our method is flexible and adaptive compared to the binary level classification.

The remainder of the paper is organized as follows. Section 2 discusses the details of the related research. Section 3 introduces the classification scheme and the MLR model. Section 4 presents the details of simulation, data processing and performance evaluation. Section 5 gives the results of the simulation experiments. Section 6 concludes the paper, with suggestions for future work.

## 2 Related work

In spite of the large number of proposed caching schemes, traffic analysis reports and replacement algorithms in the literature, only a few of them pay attention to the web object classification. (Foong et al. 1999) proposes a binary logistic regression model to predict the future access pattern. Their study considered size, type, previous hits and time since the last access as predictive variables ( $X$ s) and the response variable ( $Y$ ) maps into two categories; the object will be re-accessed in a given forward looking window (with  $Y = 1$ ), or not (with  $Y = 0$ ). This approach, though simple in design, does not address the download latencies of the object. As there is no specific object classification scheme adopted, this model is limited for content distribution applications.

A nonlinear model is used by (Koskela et al. 2003) to predict the value of each cache object using the features from the HTTP responses of the server, the access log of the cache and from the HTML structure of the object. Here the web objects are classified into  $class_0$  and  $class_1$ . The classification is done with respect to only one parameter: popularity. Even though the results are promising, it lacks a general classification approach and hence useful mainly for cache replacement.

An adaptive method is suggested in (Tian et al. 2002) using NN to predict the future access of web pages. This model is similar to (Foong et al. 1999), except for the usage of the NN. This approach has definite advantages of the NN, but lacks a strong classification scheme. The Neural Network Proxy Cache Replacement (NNPCR) technique suggested in (Cobb and ElAarag 2008), extracts frequency, recency and size from the web logs to predict the cache-ability of an object. Even though this model has high learning rate and prediction accuracy, it does not consider the previous latencies and type of the objects, which are important in the current web cache. Though the approach in (Bian and Chen 2008) is promising as it uses a page grading mechanism, its prediction mechanism is weak. This model also does not use any object classification method.

It is worth mentioning the classification scheme proposed by (Pallis et al. 2007), even though it is applicable for CDN's (Content Delivery Networks) only. A binary scheme classifies the objects as dynamic and static according to the user's interest (or popularity). However, this scheme does not consider the size, type and latency in order to estimate the quality value of the object.

Thus, the number of caching schemes based on web object classification is low and are with some limitations. So there is a need for further research in this direction. In the proposed work, we use a discrete choice parameter as a key for classification in contrast to the binary classification

in the above works. Our technique is similar to (Foong et al. 1999) in the sense that both use Logistic Regression for the object classification. Rather than concentrating *much* on the cache-ability of objects, we focus on a general, predictive object classification scheme, which can be utilized for cache admission, replacement and pre-fetching.

### 3 The proposed content classification scheme

Our proposed scheme is derived from (Foong et al. 1999), where the web access is modelled as hit/miss sequence and targeting whether an object would be referenced in the near future or not, using a binary LR model. Our idea is different from this by the use of grading parameter worthiness, which incorporates object attributes and traffic parameters to a single factor. The worthiness parameter has advantages over the other parameters such as grade (Bian and Chen 2008), cost (Li et al. 2007; Cao and Irani 2002) and popularity (Chen and Zhang 2003; Jin and Bestavros 2000) as it is comprehensive and generic. So, the worthiness parameter can be tuned for a specific form of content distribution (e.g., multimedia caching and object pre-fetching). Rather than finding the value of the worthiness of objects individually, a relationship is to be obtained among the object attributes and traffic characteristics, which is to be mapped to multilevel classes.

As mentioned earlier, caching schemes are generally influenced by four factors (characteristics) of objects while applying a cache admission/replacement strategy (Podlipnig and Böszörményi 2003) as listed below:

- *Frequency* the number of references to an object in the past (popularity).
- *Recency* the time elapsed since the last reference to the object (reflects temporal locality in the request stream).
- *Size* the size of the object.
- *Cost* the cost of fetching an object from the origin server (usually a function of frequency and size).

We consider three more parameters to estimate the worthiness of an object. They are—Consistency in popularity—the mean popularity obtained by considering the past accesses as number of windows.

- *Distance to the origin server* (or previous download latency): the time taken for an object to get into the cache. Usually this is  $p_d + S_d$ , where  $p_d$  is the propagation delay (travelling time) and  $S_d$  is the processing time at the origin server. We understand that this parameter is very difficult to measure correctly in the practical situations. However, we include this parameter to avoid caching of objects from nearby, fast, and powerful web servers.

- *Type of the object*: the different types of web objects include HTML, image and applications.

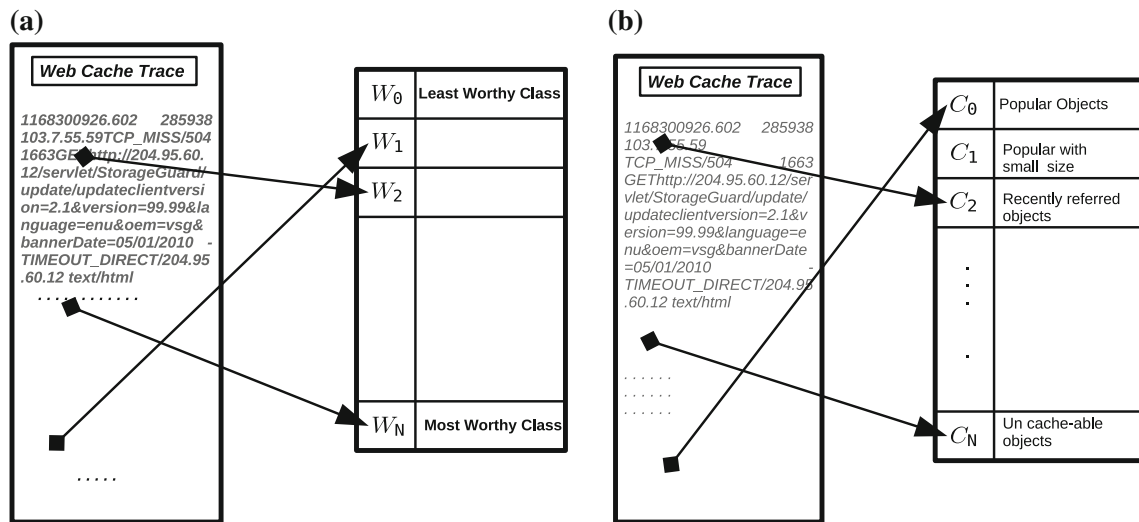
#### 3.1 The classification problem

The classification problem is essentially a decision on the class membership (Dreiseitl and Ohno-Machado 2002). More specifically, a classifier  $h$  maps any object  $x \in X$  to its true classification label  $y \in Y$  defined by some unknown function  $g : X \rightarrow Y$  on a dataset  $T = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ . The class label values could be dichotomous ( $y$  values are either 0 or 1) or polytomous ( $y$  takes more than two values) and  $x_i$  are usually  $m$ -dimensional vectors (Dreiseitl and Ohno-Machado 2002).

To classify the web objects, two approaches are possible as can be seen from Fig. 2a and b. One is by labeling the classes according to the grading parameter (e.g., *cost*, *rank* or *worthiness*), which can be used as a key for caching tasks like admission or replacement. The second approach is by assigning class label values with respect to object attributes like frequency, recency and size (e.g., a class with popular objects with small size). Both the approaches are relevant in its own way; the first approach is useful when the intention is to develop caching schemes by a single quality factor of the objects and the second approach has the advantage of grouping the objects with combination of attributes. The second approach is useful to analyze the nature of objects which are handled by the cache. We choose the first approach as our objective is to develop a caching scheme using classification.

##### 3.1.1 Object worthiness

The notion of object worthiness is similar to rank (Chen and Zhang 2003; Jin and Bestavros 2000), cost (Li et al. 2007; Cao and Irani 2002) and page grade (Bian and Chen 2008) but is different in definition and in the method of computation. The above parameters classify the objects according to cache-ability, popularity, size and cost of fetching and captures these parameters online. Their objective is to increase the performance of the cache server by keeping most of the popular objects in the cache. However, it is not feasible to keep track of the history of all these factors simultaneously. Hence most of the existing grading mechanisms have considered only the popularity and size and did a trade off between HR and BHR. This may improve the cache performance from the users perspective, but the performance will not be consistent with the changes in the traffic patterns. Our objective is to classify the web objects using the worthiness parameter which adapts to the traffic conditions and object properties. The worthiness of an object



**Fig. 2** Possible ways of classifying web objects for caching **a** Objects are classified according to a grading parameter worthiness, **b** classified according to attributes

is essentially its merit to become a new member of the cache store, or to continue its membership.

**Definition 1** The object worthiness  $W$  of a web cache is a discrete choice variable which is defined as

$$W = F(\omega_1, \omega_2, \dots, \omega_m) \tag{1}$$

where  $W$  takes values from 0 to  $N - 1$  in a  $N$ -class classifier and  $\omega_i$  s are the individual worthiness contributions. In this paper, six factors are considered for computing the aggregate worthiness as given in Tables 1, 2. The number of such independent variables is flexible and can be decided based on the relevance.

### 3.1.2 Computing the object worthiness

The goal of the data stream classifier is to go online and adapt with the line conditions. The desirable features of such a classifier are fast response time and the use of less memory and computational resources (Attar et al.

2010). Another desirable aspect of the classifier is its stability in prediction accuracy. It is known (Miller 2002; Ahn et al. 2007) that a single classifier model performs poorly. Averaging or ensembling of multiple models can address this problem (Friedman et al. 2000; Ahn et al. 2007).

Ensembling addresses the stability issue of the classifier by combining multiple classification models and features of the data. The different options of combining are parallel combining, stack combining and weak combining (Ahn et al. 2007). In weak combining method, classifiers of the same type are trained using the same set of features or subsets of the same set. In our proposed method, we use a weak combining ensembling since the classifier is meant for classifying input data stream of the web cache server. The weak combining is desirable because it learns faster and consumes less resources (Chu et al. 2004).

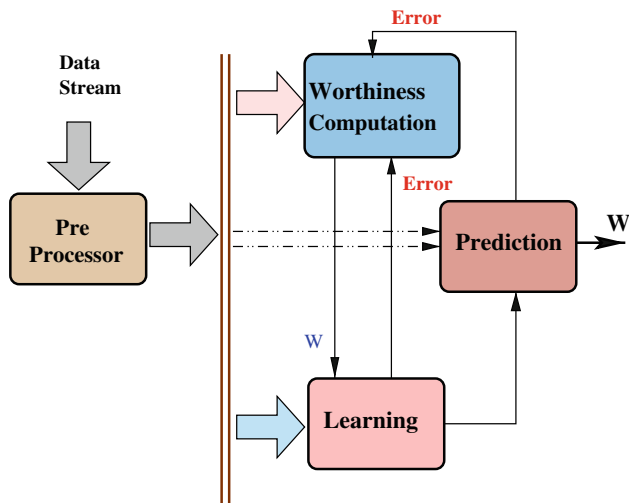
Consider a multiclass classification environment where the training data for an ensemble is represented as

**Table 1** Variables and descriptions

Variable name	Notation	Description
Independent variables		
Primary variables		
Popularity	$\omega_p$	The factor represents relative popularity of an object
Recency	$\omega_r$	The factor represents the recency of the object
Object size	$\omega_s$	Size of the object in bytes
Auxiliary variables		
Popularity consistency	$\omega_{pc}$	The factor represents bursts in popularity
Delay	$\omega_d$	The factor represents delivery time of objects or distance to the origin server
Type of object	$\omega_t$	Object types fall into three categories html/text, image/video, application
Dependent variable	$W$	Aggregate worthiness factor

**Table 2** Notations used in the model

Notations used	Meaning
$N$	Number of references
$O_i$	$i$ th cache object
$S$	Size of the object
$t_0$	Time of consideration
$t_r$	Time of last reference to the object
$f$	Frequency; number of times the object referred in the previous window
$p$	Aggregate popularity index
$k$	Number of windows
$p_i$	Popularity indices of windows, where $i = 1, 2, \dots, k$
$t_e$	Elapsed time for the object, when it is delivered from origin server
$T$	Constant used to subdue the recency factor. The value is chosen by observing the timing information
Speed	Bit rate of outer link from the cache server



**Fig. 3** A generic model for computing the worthiness from the data stream and integrating with the classifier

$$(X, Y) = \{(x_i, y_i); i = 1, \dots, N\} \tag{2}$$

where  $x_i$  is a vector-valued sample attribute and  $y_i \in \{1, \dots, m\}$  is the class label. Since  $y_i$  is mapped as new parameter worthiness (which is not directly available from the cache logs), it is computed using the adaptive technique depicted in Fig. 3. The worthiness factor is (re) computed: (1) for training the classifier, (2) when learning or prediction accuracy is not within the expected limit, (3) when a concept drift is detected.

Consider a continuous data stream with the point of observation at  $T$ . Let  $W_F$  and  $W_B$  are the forward and backward looking windows with number of samples as  $N_F$  and  $N_B$ , respectively. The simple way to compute the worthiness factor is by comparing the actual values in the  $W_F$  with the

threshold values by assigning the weight for individual worthiness contributions. We denote  $\gamma_k^l$  as the score of sample  $k$  at  $T$  for the attribute  $l$ . Then the new score of the sample  $k$  (for the attribute  $l$ ) at  $T + 1$  is  $\gamma_k^l \pm \delta_k^l$ , where  $\delta_k^l$  is the score in  $W_F$ .

Let  $A_l; j = 1, \dots, p$  be the threshold values of different  $p$  attributes and  $\alpha_l$  be the estimate of observed values in  $W_F$ .  $\delta_k^l$  is computed using a function (either growing or decaying) according to the observed value in  $W_F$  and the history of past samples as

$$\delta_k^l = \Gamma_1(N_F, \alpha_l, \gamma_k^l) \} \alpha_{(l_j)} \geq A_{l_j} \tag{3}$$

or

$$\delta_k^l = \Gamma_2(N_F, -\alpha_l, \gamma_k^l) \} \alpha_{(l_j)} < A_{l_j} \tag{4}$$

$\Gamma_1$  and  $\Gamma_2$  are chosen by analyzing the trace data and the nature of the attribute. The aggregate score  $y_k$  is computed as

$$y_k = \sum_{\forall l} \gamma_k^l \times x_k \tag{5}$$

This method computes the worthiness class of the objects by computing the score (contribution) of all attributes, successively. Minor changes and tuning are necessary for computing the score of attributes like consistency in popularity and the delay. Algorithm 1 describes this computational process.

**Algorithm 1:** Computing the worthiness factor for training

```

/* Procedure to be invoked for computing
worthiness factor; either for initial training,
upon error and concept drift */
Data: sample vector S of data stream
; /* worthiness contributions */
Output: response variable y_k
; /* worthiness class */
forall x_k in S do
  read \gamma_k^l of W_B ; /* initialize the score from
the previous window */
  foreach attribute l do
    estimate \alpha_l_j of W_F
    ; /* check the contribution in forward
looking window */
    if \alpha_l_j > A_l_j then
      \delta_k^l = \Gamma_1(N_F, \alpha_l_j, \gamma_k^l) ; /* boost the
score using the estimate of past and
future samples */
    end
    else
      \delta_k^l = \Gamma_2(N_F, -\alpha_l_j, \gamma_k^l) ; /* reduce the
score to place in lower class */
    end
    \gamma_k^l = \gamma_k^l \pm \delta_k^l ; /* new score
corresponding to the attribute l at
T + 1, \pm is applied according to the
nature of attribute */
  end
  y_k = \sum_{\forall l} \gamma_k^l \times x_k
end

```

### 3.2 The MLR model

A MLR model is used for the data where the dependent (or response) variable is unordered or polytomous (multicategory), and the independent (or explanatory) variables are continuous or categorical predictors (Hosmer and Lemeshow 2000; Agresti and Wiley 1990; Wang 2005). This type of model is therefore measured on a nominal scale and was introduced by McFadden in 1974. This model is also called as *discrete choice model* and is an extension of the binary logistic regression. In a binary logistic model, a dependent variable has only a binary choice (e.g., presence or absence of a characteristic), whereas the dependent variables in a multinomial logistic regression model can have more than two choices that are coded categorically, and one of the categories is taken as the reference category (Wang 2005).

An important application of the multinomial logistic model is in determining the effects of explanatory variables on a subject’s choice from a discrete set of options. An explanatory variable takes different values for different response choices.

A general MLR model is of the form

$$Y = \beta \times X \tag{6}$$

where  $Y$  is the unordered dependent variable,  $\beta$  is the vector of coefficients and  $X$  is the vector of independent variables.

We assume that the categories of outcome or response variables,  $Y$ , are coded as  $0, 1, \dots, J + 1$ , and the explanatory variables  $X$  is a vector of size  $p + 1$ , as in (Hosmer and Lemeshow 2000; Agresti and Wiley 1990). In the model we define  $J$  logit (logistic) functions  $g_1(x)$  to  $g_J(x)$  as follows:

$$\begin{aligned} g_1(x) &= \log \left[ \frac{P(Y = 1|x)}{P(Y = 0|x)} \right] \\ &= \beta_{10} + \beta_{11}x_1 + \beta_{12}x_2 + \dots + \beta_{1p}x_p \\ &= X\beta_1 \end{aligned}$$

$$\begin{aligned} g_2(x) &= \log \left[ \frac{P(Y = 2|x)}{P(Y = 0|x)} \right] \\ &= \beta_{20} + \beta_{21}x_1 + \beta_{22}x_2 + \dots + \beta_{2p}x_p \\ &= X\beta_2 \end{aligned}$$

$$\begin{aligned} g_J(x) &= \log \left[ \frac{P(Y = J|x)}{P(Y = 0|x)} \right] \\ &= \beta_{J0} + \beta_{J1}x_1 + \beta_{J2}x_2 + \dots + \beta_{Jp}x_p \\ &= X\beta_J \end{aligned}$$

The conditional probabilities are now calculated as

$$P(Y = i|x) = \frac{\exp(g_i(x))}{\sum_{k=1}^J \exp(g_k(x))} \tag{7}$$

where  $Y_i$  is the unordered categorical dependent variable for the observation  $i$  (which takes an integer value from 1 to  $J$ ),  $x_i$  is the vector of  $k$  explanatory variables for observation  $i$ , and  $\beta_j$  is the vector of coefficient for category  $J$ .

We apply the multinomial logistic regression by considering the object worthiness  $W$  as a dependent variable and the independent variables ( $\omega$ ’s) are drawn from the cache logs. Thus,

$$W = \beta \times \Omega \tag{8}$$

where  $\Omega$  is the vector of  $\omega$  values. The object worthiness is defined to have eight<sup>1</sup> categories. The ‘0’ category is the least worthy objects and the category ‘7’ is with the most worthy objects. Note that we could use Ordered Logistic Regression (OLR) since the dependent variable object worthiness has some order. But the MLR method is robust and covers a much wider class, and the OLR method is treated as a special case of MLR (Hosmer and Lemeshow 2000; Komarek 2004).

The worthiness factor of web objects is an unknown function. A MLR model is constructed and trained to approximate this unknown function from the real-world data. The model is trained using  $\omega$  values to estimate  $\beta$  coefficients. The beta values are computed by the Maximum Likelihood Estimation (MLE)<sup>2</sup> method (Hosmer and Lemeshow 2000; Agresti and Wiley 1990). Then the model is validated using an internal validation technique. Figure 4 shows the MLR training and prediction. To train the model,  $\omega$  values are fed along with aggregate worthiness value  $W$ . This yields a matrix of  $\beta$  coefficients. The details of validation and testing are discussed in subsequent sections.

### 4 Simulation

Our simulation experiments include, processing of the web traces, training and verification of the MLR model and the cache simulation. We have collected the traces and sanitized logs from the IRcache proxy-cache servers (NLNLR, Last accessed in January 2010) as shown in Table 3a. These logs typically contain 10 fields as shown in Table 3b. The hit ratio in Table 3a is the hit ratio measured

<sup>1</sup> There are six explanatory variables. Hence, we consider the nearest power of 2 as 8. This gives a flexibility to redefine  $W$ , according to an application.

<sup>2</sup> Maximum likelihood estimation begins with writing a mathematical expression known as the Likelihood Function of the sample data. The likelihood of a set of data is the probability of obtaining that particular set of data, given the chosen probability distribution model. This expression contains the unknown model parameters. The values of these parameters that maximize the sample likelihood are known as the Maximum Likelihood Estimator.

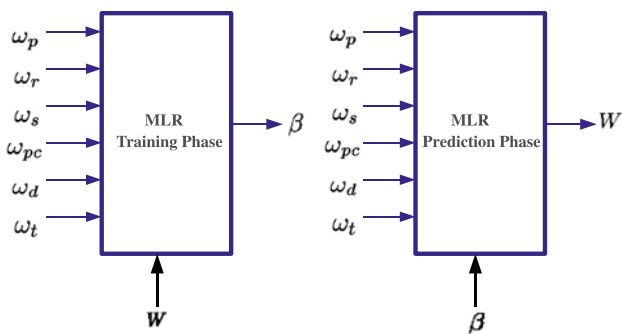


Fig. 4 MLR training and prediction

using a standard trace analyzing tool *proxytrace* and Zipf slope is measured using *zipfR* package of R Language software (Team 2008).

The initial file processing is done using a standard Unix utility *awk*, to extract the timestamp, frequency, size, URI, and the type of the objects. Then, the individual worthiness contributions are computed, and normalized to generate the dataset.

#### 4.1 Data pre-processing

The need for preprocessing arises because of the large dispersion of data of the collected trace files. For example, size of an object varies from few bytes to several Mega bytes and the timing informations used to be much more scattered. The data pre-processing serves the following purposes: (i) subdues the data so that it can be fed directly to the MLR model, (2i) computes the worthiness contribution of each data (field). The worthiness contribution of the object’s popularity (Jin and Bestavros 2000) is computed as a relative score. Let object  $O_i$  be referenced  $f_i$  number of times at an instant. Assume that there are  $n$  distinct objects referred through the cache (Refer Table 2). Then the worthiness factor of  $O_i$  contributed by popularity is computed as

$$\omega_p \approx \frac{f_i \times n}{N} \tag{9}$$

Further, the contribution of the object’s size is considered as an integer function of its size and speed of the outer link (in our experiments, we assume its value as 256 Kbps to 2 Mbps, depending upon the average size of the objects in the workload) and is computed as

$$\omega_s \approx \frac{S}{Speed} \tag{10}$$

To capture the temporal locality, the worthiness contribution of recency of an object is evaluated as shown in Fig. 5. Let the time of last access to the object be  $t_r$ . Let the elapsed time (or number of references to other objects) since its last access be  $\Delta\tau$  which is equal to  $t_0 - t_r$ . Then the contribution to the worthiness factor is

$$\omega_r \approx \frac{\Delta\tau}{T} \tag{11}$$

The type of an object is classified into four categories; HTML (php, text), image/video, others (pdf, ps) and applications (cgi, asp). The type ‘applications’ include those objects, which are dynamically generated in the

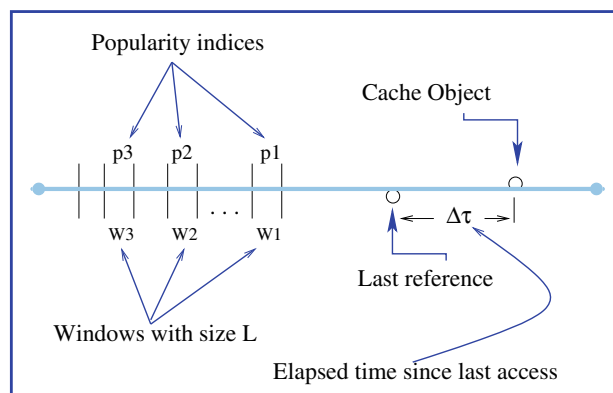


Fig. 5 Computing the popularity consistency and the weightage of recency

Table 3 Traces used and the fields of the cache logs

Date	Number of requests	Trace/source	Hit ratio (%)	Zipf slope
(a) Traces used				
1st September 2007	2,80,062	NLANR-pa	36	0.72
1st October 2008	2,10,734	NLANR-pa	32	0.76
10th August 2007	2,43,356	NLANR-bo2	26	0.69
9th January 2007	5,44,356	NLANR-sj	20	0.80
23rd January 2005	28,338	Internet Traffic Archive	48	0.72
(b) Fields in the cache log				
Time stamp	Elapsed time	Client IP	HTTP code	Size
Request Method	URL	User ident	Hierarchy data	Content type



origin server. Hence these objects are not good candidates for caching. Weightages are assigned to the different types following a similar convention as in (Tian et al. 2002). These weights are needed as the variable *type* is non-metric. We choose ‘15’ for HTML, ‘10’ for images, ‘5’ for other types and ‘0’ for applications. These values are chosen based upon the relative frequency of the type of files.

The consistency in popularity is taken into account to reduce the effect of bursts (large number of accesses in a small duration) in popularity (Jin and Bestavros 2000). In Fig. 5, the object references are taken as number of windows. Let the object  $O_k$  be referred at  $t_0$  and  $p_i, i = 1, 2, \dots$  be the popularity indices with  $p$  be the mean popularity (weighted average of  $p_i$ 's). We need to check whether the current popularity is due to a burst in one of the windows. For this purpose, we do the variance analysis of  $p_i [\sigma(p_i)]$ . That is, the integer values of the range [0 : 9] are assigned to  $\omega_{pc}$  with the following restrictions:

- maximum value 9 is assigned if the current popularity is consistent ( $p_i \approx p, \forall i$  or  $\sigma(p_i) \approx 0$ ).
- 8 is assigned if  $p$  is less consistent ( $p_i > r_{approx}, \exists i$ ) and so on.
- minimum value 0 is assigned if there is a burst in one of the windows ( $p_i \neq p, \exists i$  or  $\sigma(p_i) \approx 1$ ).

The factor  $\omega_d$  is obtained by measuring the elapsed time of the object when it is delivered from the origin server. An object which is delivered faster is a less worthy object for the cache. The same object may experience varying elapsed times. This situation is handled by considering the mean of the elapsed times. Finally, the fields of the dataset ( $X$ ) are normalized using a linear normalization function in the range  $[X_1 : X_2]$  as

$$X = \frac{X - X_{min}}{X_{max} - X_{min}} \times (X_2 - X_1) + X_1. \tag{12}$$

### 4.2 Simulation metrics

The Correct Classification Rate (CCR) is a measure of classification accuracy. CCR gives the percentage of inputs that are correctly classified. However, CCR alone may be insufficient for measuring the classification accuracy, especially if the data is imbalanced (Xu et al. 2005). Therefore, Misclassification Rate (MCR), True Positive Rate ( $T^+$ ), True Negative Rate ( $T^-$ ), geometric mean ( $G_{mean}$ ) and  $e_n$  are also used in the the MLR model as given in Table 4.

The web cache performance is measured using the metrics, HR and BHR. HR gives the percentage of number of requests served from the cache server and BHR gives the percentage of volume in bytes, served. For using the object worthiness as a key for the classification, we define two

**Table 4** Additional metrics used

(a) Metrics used		
1.	$T^+ = \frac{TP}{TP+FN}$	
2.	$T^- = \frac{TN}{TN+FP}$	
3.	$G_{mean} = \sqrt{T^+ \times T^-}$	
4.	$e_i =$ Depth of error, with $i$ being the number of classes shifted when the classification is wrong	
	Predicted positive	Predicted negative
(b) Confusion matrix		
Actual positive	True positive (TP)	False negative (FN)
Actual negative	False positive (FP)	True negative (TN)

new performance metrics named WHR (worthiness hit ratio) and WBHR (worthiness byte hit ratio) for the simulation experiment. The definition of WHR is analogous to the performance metrics used by (Gonzalez-Canete et al. 2006) and is given as

$$WHR = \frac{R_{Hit} X R_{mw}}{R_N R_w} \tag{13}$$

where  $R_{Hit}$  is the total number of objects served from the cache,  $R_N$  is the total number of requests,  $R_{mw}$  is the number of the most worthy objects served from the cache and  $R_w$  is the total number of worthy objects including class 0. When all objects are worthy, then WHR will be reduced to HR. Similarly, WBHR is defined as

$$WBHR = \frac{B_{Hit} X B_{mw}}{B_N B_w} \tag{14}$$

The terms in the above equation have similar meaning as in Eq. 13, except for the word ‘object’ used is replaced with the word ‘byte’. Another important cache performance metric is access time or response time which is the time taken for the user to access a web object. We measure the mean response time (MRT) for different cache sizes, HR and WHR to check the effectiveness of the classification scheme.

### 4.3 Performance evaluation

Before evaluating the performance of the scheme, the model is validated using the bootstrapping method. The validation process is necessary to make sure that the converged model is not simply the artifact of the data used. In the bootstrapping method of validation, sub samples are used for the simulation (Efron and Gong 1983; Steyerberg et al. 2001). The dataset is divided into  $m$  blocks of equal size. Then the simulation is performed  $m$  times, each time exactly leaving one block. This method works better than the cross-validation technique as reported in (Steyerberg et al. 2001). The performance evaluation of the MLR

model is a testing process to make sure that the model generalizes the relationship between predicative inputs and the response.

We implement the classification scheme in an object replacement method called LRU-M (Least Recently Used-Multilevel Classes). This method is similar to the LRU-C (Koskela et al. 2003) except for the use of multilevel classes to arrange the web objects in the cache store. In LRU-M, the replacement of objects is carried out from a designated class and the classes below that. For example, when the cache is full, an object from  $class_2$  and the classes below that, is selected for eviction (in reverse order). If the requested object is not found in the cache, then the class of the object is checked. If the class, is say 2, then the object is placed at the bottom of the segment  $m_2$  and if the object is found in the cache then the object is placed at the top of the segment  $m_2$ . The Objects are always removed from the bottom of the stack segment of a particular class. LRU-M performance needs to be compared with that of the conventional LRU and LRU-C<sup>3</sup> in terms of WHR, WBHR and response time of the objects.

## 5 Results

We use the open source R Language statistical software (Team 2008) and Zelig package (Imai et al. 2006) to implement the MLR model. The dataset required for the estimation is generated from the traces of IR Cache (NLNR, Last accessed in January 2010). The training and the testing process is shown in Fig. 4. Several MLR forms were experimented to fit the data. Initially, all the input variables (worthiness contributions) are considered as primary variables. In order to check the fitness or convergence of the data the  $p$ -value is checked. As this resulted in a less accurate fitting (with high  $p$  value) or non-convergence, two input variables ( $\omega_{pc}$  and  $\omega_d$ ) are fed as auxiliary variables.

### 5.1 Classification

#### 5.1.1 Training and validation

The model is initially trained using a dataset with 5,000 records (required for convergence). The  $\beta$  value are computed using the Maximum Likelihood Estimation (MLE) method and Pearson residual values are used to confirm the convergence of the iterations. Using the  $\beta$  values,

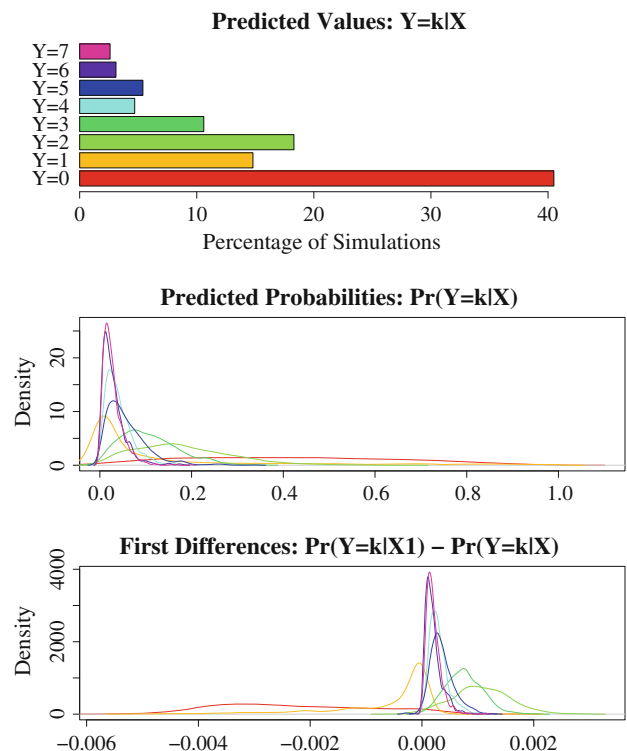
<sup>3</sup> Here, we implement LRU-C method using binary LR method. Hence, worthiness factor will have only two classes;  $W = 0$  and  $W = 1$ . Also, we do not consider the features from HTTP responses of the server and the HTML structure of the object.

probabilities and the values of the dependent variable  $W$  are computed and are plotted in Fig. 6. In one of the experiments with a dataset size of  $10^4$ , we observe that  $class_0$  ( $Y = 0$ ) and  $class_1$  ( $Y = 1$ ) together constitute about 52% of the objects. Note that  $class_0$  (40%) contains objects which are not worthy in any way; either these objects are not cache-able or have very minute value of popularity, size and recency. This shows that, any caching scheme should equip with a consistency handling mechanism to achieve higher HR.

In the bootstrapping method, a total of 50 sample sets of the same size of the original data sets were utilized. Table 5 compares the results of simulation and validation. All the independent variables consistently correlate with the worthiness factor. This shows that the model has good prediction power and is not simply the artifact of the data used to develop the model.

#### 5.1.2 Accuracy

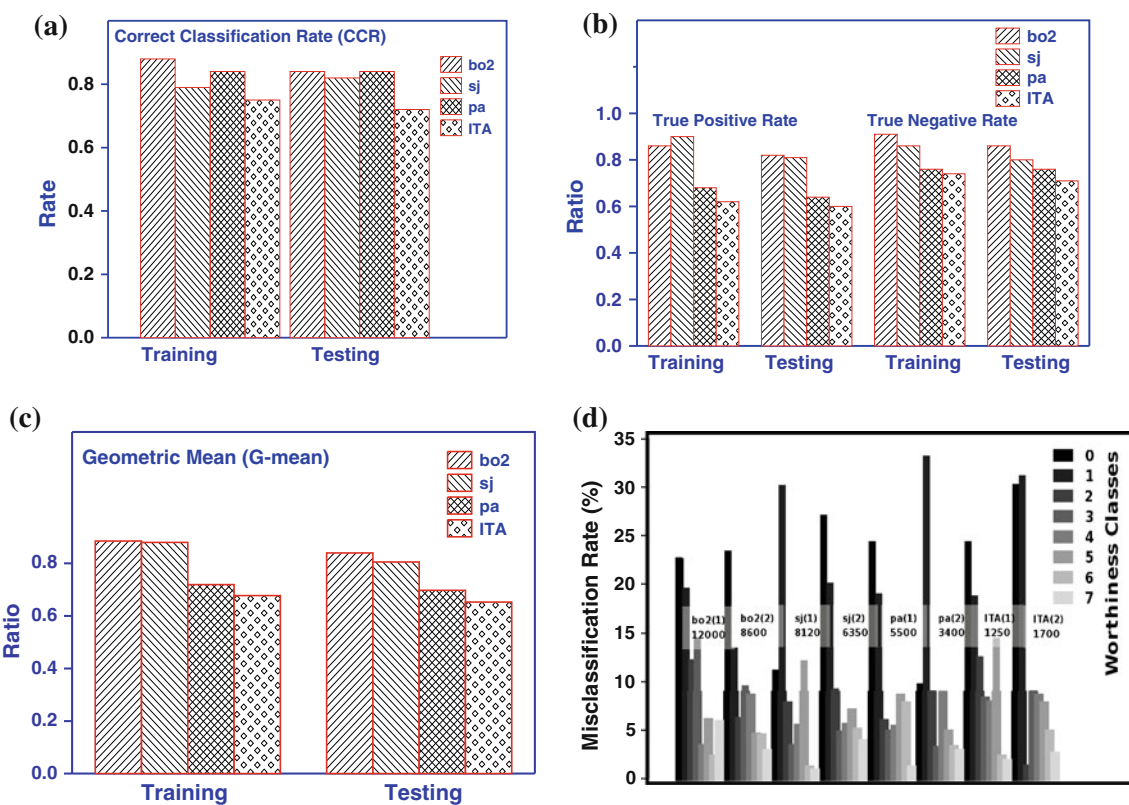
Figure 7a–c show Correct classification Rate (CCR), True Positive Rate ( $T^+$ ), True Negative Rate ( $T^-$ ) and  $G_{mean}$  ratio of different datasets for training and verification. The highest CCR observed for training (88%) and testing (84%) is with the dataset of *bo2.ircache.net*. This bo2 dataset performs consistently for training and testing of  $T^+$  (83%,



**Fig. 6** Worthiness of cache objects. **i** Values of classes, **ii** probabilities, **iii** first differences.  $Y, k$  and  $X$  are analogous to  $W, i$  and  $\omega$

**Table 5** Probabilities for simulation and validation

	Simulation				Validation			
	Mean	SD	2.5%	97.5%	Mean	SD	2.5%	97.5%
Pr(Y = 0)	0.470024	0.193630	0.318606	0.621442	0.451072	0.176230	0.306656	0.595488
Pr(Y = 1)	0.267541	.070040	0.186914	0.348168	0.275239	0.214592	0.016593	0.384547
Pr(Y = 2)	0.092402	0.005900	0.056441	0.128363	0.096156	0.054065	0.018219	0.174093
Pr(Y = 3)	0.065546	0.004803	0.012829	0.118279	0.068462	0.044889	0.014413	0.122511
Pr(Y = 4)	0.022030	0.001924	0.003435	0.040625	0.019347	0.014138	0.003271	0.035424
Pr(Y = 5)	0.039428	0.003559	0.006182	0.072674	0.040871	0.032663	0.007219	0.074522
Pr(Y = 6)	0.018898	0.001326	0.002528	0.035268	0.019293	0.017164	0.002271	0.036315
Pr(Y = 7)	0.007006	0.006278	0.000674	0.013338	0.007978	0.006408	0.000612	0.015345



**Fig. 7** **a** Correct Classification Rate of different datasets. **b** True Positive Ratio ( $T^+$ ) and True Negative Ratio ( $T^-$ ). **c**  $G_{Mean}$  ratio. **d** Misclassification rates

81%),  $T^-$  (89%, 83%) and  $G_{mean}$  (90%, 83%). These results indicate that the model had been trained well and is doing good prediction.

5.1.3 Error rates

Figure 7d shows the misclassification rates for the eight classes of datasets. Two datasets generated from each trace are used for this check. Random number of samples are used in every test. The general observation is that,

misclassification rates are higher in lower classes. For  $class_0$ , it is 12–25% and for  $class_1$ , the error rate is 9–31%. The misclassification rate ranges from 1 to 5% for classes 5–7. This observation is maintained for all datasets. The statistical package Zelig (Imai et al. 2006) does not have a direct measure of accuracy. Hence, we measure its accuracy manually. The model is trained using 5,000 samples. In order to test the prediction accuracy, we used another data set of size 100, 500, 1,000, 2,000, 5,000 and 10,000 samples. We found that the prediction accuracy varies from

**Table 6** Misclassification and prediction error

Samples	Correct	Accuracy (%)	$e_1$ (%)	$e_2$ (%)	$e_3$ (%)	$e_4$ (%)	$e_5$ (%)	$e_6$ (%)	$e_7$ (%)
100	56	56	76	18	4	2	0	0	0
500	312	62.4	74	16	6	3	1	0	0
1,000	611	61.1	84	11	2	3	0	0	0
3,000	1,923	64.1	79	19	2	0	0	0	0
5,000	3,567	71.3	69	24	5	1	1	0	0
10,000	7,862	78.6	73	23	2	2	0	0	0

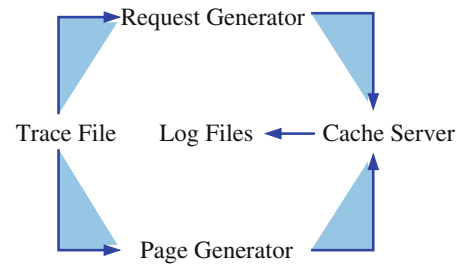
56 to 78.6%. Further, we investigated the depth of error when a prediction fails. We denote these prediction errors by  $e_i$ , where  $i$  is the number of classes shifted, when a prediction fails. We found that in most of the cases (as indicated in Table 6) the prediction shifts by only one class. We are sure that this error may not have a significant impact on the caching policies.

## 5.2 Cache simulation

We have customized the simulation tool *WebTraf* to meet our requirements (Markatchev and Williamson 2002). The LRU stack model is modified to hold multiple class data. The stack is divided into segments dedicated for  $class_0$ ,  $class_1$  and so on. Also, the LRU method is modified by adding the class label to the *object id*. In order to generate the dataset from the trace files, *traceconv* (Tracegraph 2005) is used. The *traceconv* generates two datasets; request stream generator (which preserves the temporal locality) and page pool generator. We use a standard simulation setup with values as shown in Fig. 8 and Table 7, respectively. The simulation is repeated for LRU, LRU-C and GDSF for two differently trained MLR model. The cache simulations were carried out, with varying cache size (100 MB to 4 GB). We observe that the dataset performance in training and testing are more or less getting repeated in the cache simulation also (with certain exceptions).

### 5.2.1 HR and BHR

Figure 9a and b show the performance comparisons for different cache sizes such as 256 MB, 512 MB, 1 GB and 2 GB, respectively. In terms of HR, LRU-M has a slight advantage over LRU-C and GDSF. LRU-M performs much better in terms of BHR compared to the other two. With 2 GB of cache size, LRU-M is much superior compared to all other methods. Here, GDSF performs better than LRU-C. This could be due to the fact that LRU-M replaces the objects based on the worthiness factor, whereas GDSF uses a combination of size and

**Fig. 8** Simulation setup**Table 7** Simulation environment values

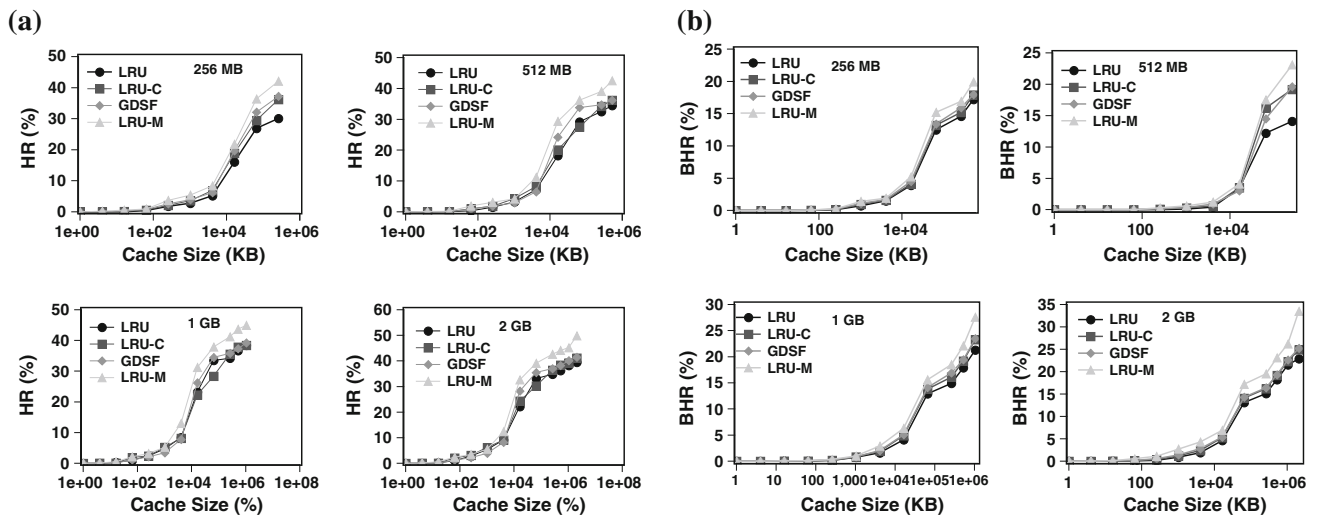
Cache server	Single
Number of web servers	100–10,000
Web cache size	512 MB–4 GB
Cache size as % of web server's size	10–80%
Number of requests	5,000–1,00,000
Zipf parameter of input trace	0.72–0.84, heavy tailed
Number of unique requests	2,000–10,000
Policies tested	LRU, LRU-C, GDSF and LRU-M

popularity. Note that the worthiness factor compounds the effect of multiple parameters.

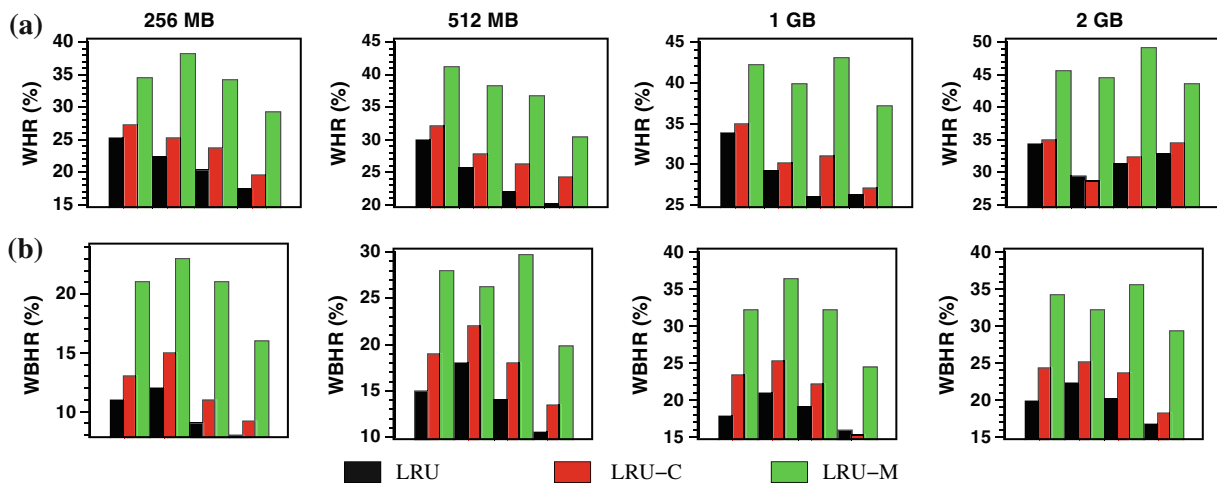
### 5.2.2 WHR and WBHR

Figure 10a and b show the performances of LRU, LRU-C and LRU-M. It is observed that LRU-M performs better for all datasets considered. Here, the simulation is performed by replacing  $class_0$  objects and the worthiness hit ratios are measured by considering the objects from  $class_1$  and above, as  $R_{mw}$ .

The performance of LRU-M is checked by replacing the objects from different classes.  $LRU/m/n$  denote the LRU-M policy with the replacement of objects allowed from the  $n$ th class and below. Figure 11a and b compare the WHR and WBHR performances. It is observed that the LRU-M methods ( $LRU/m/n$ ) perform better than the LRU and

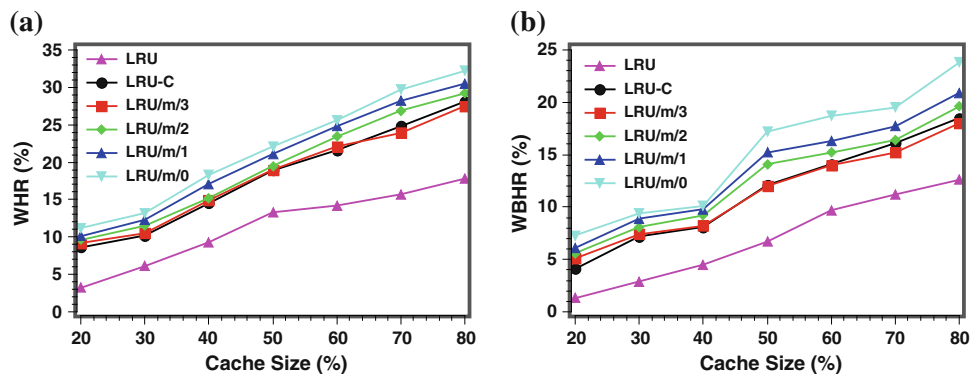


**Fig. 9** a HR comparisons of different policies (CCR = 75–85%). b BHR comparisons of different policies (CCR = 75–85%)



**Fig. 10** a WHR comparisons of different policies (CCR = 75–85%). b WBHR comparisons of different policies (CCR = 75–85%)

**Fig. 11** a WHR comparisons of different policies (CCR = 75–85%). b WBHR comparisons of different policies (CCR = 75–85%)

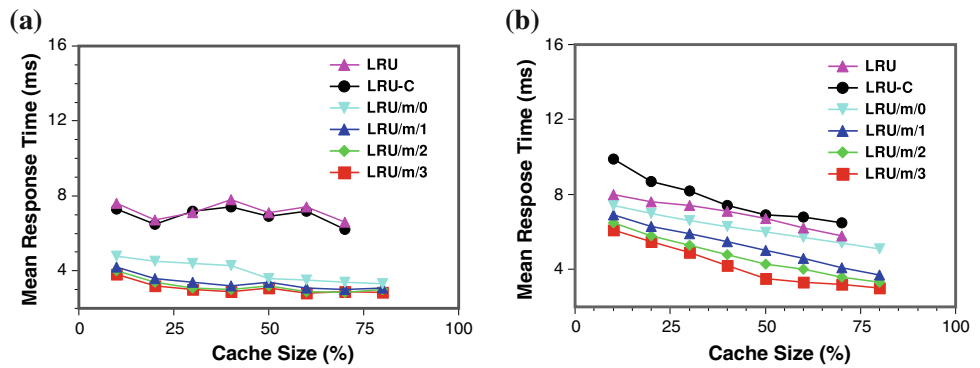


LRU-C. Note that *LRU/m/3* uses about 40% of the objects as worthy and performs almost similar to that of LRU-C. This implies that the cache size is better utilized in multilevel classification than with the binary classification approach.

### 5.2.3 MRT

The MRT is measured for different cache sizes. We compare the MRT of *bo2.ircahe.net* (best performed in classification)

**Fig. 12** Mean response time observed for binary and multi-level object classification.  
**a** Trace file of *bo2.ircache.net* with  $CCR = 0.85$ ,  $T^+ = 0.82$ ,  $T^- = 0.79$  and  $G_{mean} = 0.80$ .  
**b** Trace file of *ITA* with  $CCR = 0.71$ ,  $T^+ = 0.73$ ,  $T^- = 0.68$  and  $G_{mean} = 0.70$



and *ITA* (worst performed in classification) traces. The response is faster in LRU-M compared to that of LRU and LRU-C as can be seen from Fig. 12a and b. We now compare the response time of the *LRU/m/n* replacement methods. Though the *LRU/m/3* gives a low WHR compared to the other *LRU/m/n*, its performance is better in terms of the response time. This is because the *LRU/m/3* policy works on worthy objects of *class<sub>3</sub>* and above. This result shows that the worthiness factor is capable of capturing the latency ( $\omega_d$ ) and size ( $\omega_s$ ) information of objects better than the other grading parameters. Worthiness hit ratios are measures of worthy objects stored in the cache. From the simulation logs, it is possible to compute the response time of each object. The MRT is measured by finding the average of response time for a particular cache size and then it is mapped to the corresponding hit ratios. Table 8 shows the MRT for different HRs and WHRs. The results show that the MRT is less for WHR compared to the HR of similar value. A similar observation is found in the case of BHR and WBHR also. These results give us the confidence to state that the worthiness hit ratios are better performance indicators than the ordinary HRs.

### 5.3 Comparison with ANN

A neural network is a set of interconnected simple processing elements called neurons, where each connection has an associated weight. A neural network can achieve the desired input-output mapping with a specified set of weights (Xu et al. 2005); therefore we can train the neural network to do a particular job by adjusting the weights on each connection. A neural network with one or more hidden layer is called multilayer perceptrons. Back propagation is the commonly used method for adjusting the weights in MLP (Cobb and ElAarag 2008). Back-propagation iteratively processes the training data through input forward propagation and error backward propagation to search for a set of weights that can model the problem so as to minimize the network prediction error (Xu et al. 2005). An input vector according to Table 1, generated from the

**Table 8** Comparison of HR, WHR, BHR, WBHR and MRT

MRT	Ratio (%)									
	10	15	20	25	30	35	40	45	50	
$MRT_{HR}$	9.81	9.07	8.39	7.23	6.51	6.40	6.19	5.83	5.58	
$MRT_{WHR}$	8.05	7.91	7.52	7.04	6.05	5.73	5.16	4.64	–	
$MRT_{BHR}$	9.17	8.82	8.12	7.07	6.28	5.71	–	–	–	
$MRT_{WBHR}$	7.84	7.38	6.37	5.42	4.65	–	–	–	–	

**Table 9** ROC characteristics: area under ROC curve for different data sizes

Method	<5,000	10,000	20,000	>30,000
MLR (%)	67.2	79.2%	88.7	93.6
ANN (%)	76.7	81.4	86.3	89.2

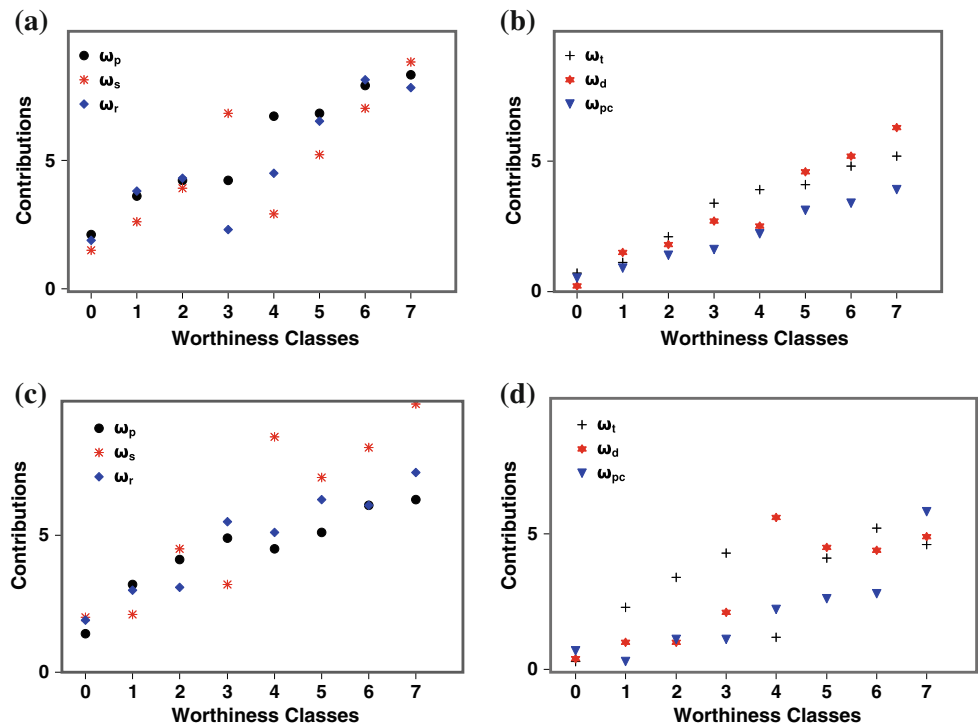
**Table 10** Comparison between MLR and ANN based caching schemes: HR performance for different cache sizes

Method	256 MB	512 MB	1 GB	2 GB
MLR based (%)	29.2	34.5	39.2	44.6
ANN based (%)	30.6	35.2	38.8	42.8

cache logs is utilized for the training session. We use a standard back-propagation technique using delta function to adjust the weights. Similar to the MLR model, the worthiness parameter is labelled (from 0 to 7). The model is validated using m-fold cross validation technique (Steyerberg et al. 2001).

A brief comparison between MLR and ANN based, classifier and caching, models is given in Tables 9 and 10. The results indicate that the MLR based scheme achieves a comparable performance with that of the ANN based methods. The ROC characteristics suggest that MLR is more suitable for classifying large quantity of web cache data.

**Fig. 13** Worthiness contributions of a sample data. **a** and **c** Popularity, size and recency. **b** and **d** Popularity consistency, delay and type of the objects



### 6 Discussion

The experimental and simulation results of classification and caching are really positive. The CCR,  $T^+$ ,  $T^-$  and  $G_{mean}$  give good accuracy rates (above 80%) and hence the classifier is said to have good prediction capability. To find the effect of worthiness contributions ( $\omega$ 's) on aggregate worthiness  $W$ , we find the average worthiness contributions of each factor in every class. These factors are normalized to a range [1:10], and is plotted in Fig. 13a. We observe that the frequency, size and recency are the most influential parameters on the aggregate worthiness as depicted in Fig. 13a and c. Among these parameters it is difficult to identify the most influential one. In general, the contributions are more or less equal in lower and upper classes, and it shows variations in the mid range classes. Among the other three parameters,  $\omega_{pc}$  is dominant in the lower classes. We could observe that this parameter alone is capable of shifting the class number from 0 to 1.  $\omega_t$ 's contribution is almost uniform in all the classes, whereas  $\omega_d$  has a slight edge in the higher classes as can be seen from Fig. 13b and d.

The overall statistical significance is to be tested with the Wald's test (as suggested by statisticians). We have not performed Wald test as MLE and the Wald test give very similar conclusions in most of the situations (Agresti and Wiley 1990). Even though the performance of the cache with respect to HR and BHR are not very promising, WHR and WBHR performance are with consistency. The performance of the cache with respect to the MRT is better for small WHRs.

This research is an attempt to apply the MLR for classifying the web objects according to the novel grading parameter worthiness, making better caching decisions. Since the proposed method is generic in nature, cache designers may adopt context specific choices. One can add or remove independent variables to redefine the worthiness factor. We suggest frequency, recency and size as mandatory parameters and the remaining three parameters are optional. Though we have demonstrated the merit of our technique through LRU-M policy, cache designers may choose any class based upon the replacement policy to suit their application. Further, any domain specific information (for e.g., rank of the web sites) can be added to refine the worthiness factor.

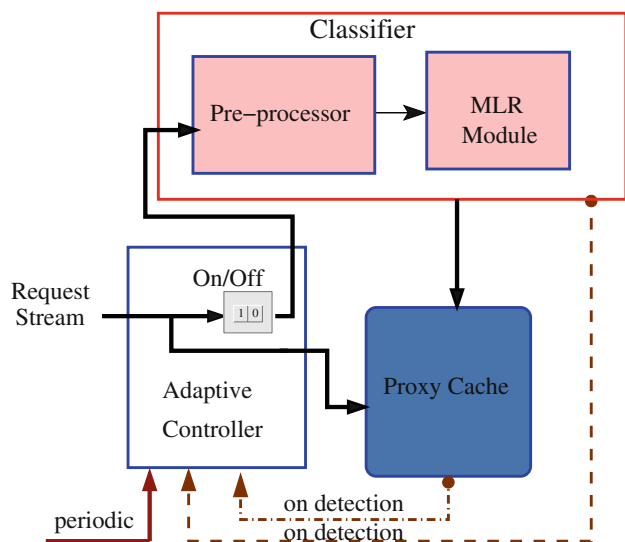
#### 6.1 Concept drift

Machine learning methods suffer from *concept drift* and *contamination* in the input data. A difficult problem with learning in many real-world domains is that the concept of interest may depend upon some *hidden context*, not given explicitly in the form of predictive features (Tsymbal 2004). Often the causes of change is hidden, not known a priori, making the learning task more complicated. Changes in the hidden context can induce more or less radical changes in the target concept, which is generally known as concept drift (Tsymbal 2004; Gao et al. 2007). In an adaptive web cache, the concept drift may occur in the data stream deteriorating the performance of the classifier and the proxy cache. Rebuilding the classifier model using

the most recent data is one solution to this problem (Wang et al. 2003).

In general, the approaches to cope with concept drift can be classified into two categories: (1) approaches that adapt a learner at regular intervals without considering whether changes have really occurred, (2) approaches that first detect concept changes, and then, the learner gets adapted to these changes (Gama et al. 2004). Difficulties arise in detecting the concept drift, as the performance reduction may also be due to the presence of *noise* (or contaminated) in the input data (Gama et al. 2004; Klinkenberg and Renz 1998). Hence, we suggest a two-tier mechanism to automate the proxy and to handle the concept drift in a running proxy cache, as depicted in Fig. 14. In this method, the adaptive controller is turned ON and OFF periodically and on detection of drift. The controller is periodically triggered for rebuilding the model. Here, the difficulty is in setting the frequency of triggering (Gama et al. 2004) and the size of the recent input data for training the classifier. A cache designer must set these values by experimentation or by applying some heuristics (Klinkenberg and Renz 1998; Gao et al. 2007). The classifier and the proxy performances shall be checked continuously to detect the concept drift (Lu et al. 2002). The parameters used for detecting the drift are HR, WHR, CCR,  $T^+$ ,  $T^-$  and  $G_{mean}$ . We do acknowledge the weakness of this method, as the performance of the classifier and the proxy may be affected by many other factors also.

We conduct experiments by inducing the concept drift to the synthetic and real data. The synthetic traces are generated using *WebTraf* (Markatchev and Williamson 2002) and *traceconv* (Tracegraph 2005) tools. Concept drifts are induced in two ways; abrupt shifts and small shifts in randomly selected blocks. The concept drift is simulated



**Fig. 14** The adaptive cache model to tackle concept drift

using the *RapidMiner* (Mierswa et al. 2006) tool. The parameter  $\lambda$  is the measure of shifts with values ranging from 0 to 1. A value of  $>0.5$  for  $\lambda$  is treated as abrupt shift and  $<0.1$  is considered to be a small shift. Cache simulation is carried out using two types of datasets; one set is with three number of abrupt shifts in randomly selected blocks and the other is with one number of abrupt shift and ten number of small shifts. The results, in terms of the cache performance, are shown in Fig. 15a and b. It is observed that the proposed model is capable of withstanding both types of shifts (with a small reduction in the cache performance).

## 6.2 Computational overhead

The MLR method is a light-weight machine learning technique, which causes less computational overhead than NN, SVM and Decision Trees. It is reported that the LR method is faster among the statistical methods and stands second in terms of accuracy criteria (Lim et al. 2000). Their training time growth is linear with respect to the number of samples (Landwehr et al. 2005). Our method requires additional computational cost for pre-processing the data and for training the LR model. The data pre-processing can be done in linear time and the asymptotic complexity for building a logistic regression model is  $O(n \cdot v^2)$  where  $n$  is the number of training samples and  $v$  is the number of attributes in the data (Landwehr et al. 2005).

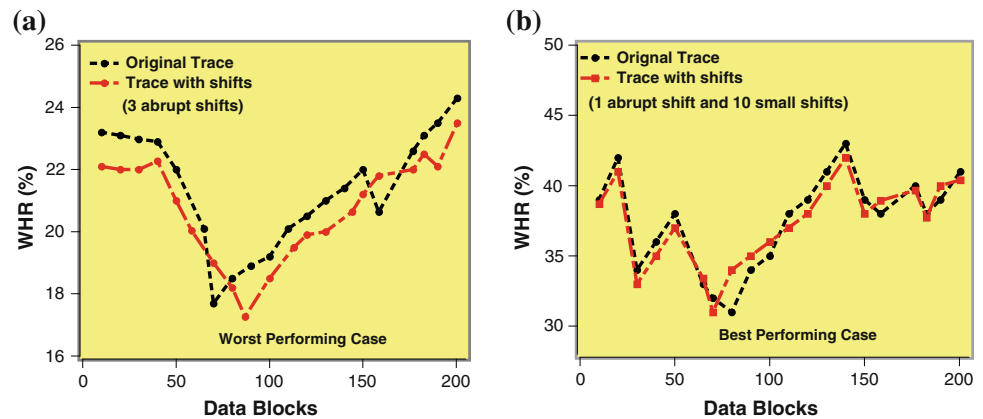
## 7 Conclusions

Web caching has a history of more than one decade. It started by drawing ideas from the processor cache, but later moved to totally different approaches. Numerous researches have been carried out in this area. Most of them are variants of LFU, LRU and GDS that rely upon the general traffic characteristics. These methods usually have some assumptions on the traffic characteristics in order to support the algorithm design. The sole advantage of such schemes is simplicity. In contrast to this, there are complex caching schemes, which employ intelligent and adaptive mechanisms utilizing ANN or genetic algorithms. Naturally, they perform much better than the basic caching techniques but with very high computational overhead.

The proposed classification scheme in this paper takes a middle path, which uses a light-weight machine learning technique. It uses a generic classification scheme with a novel grading parameter, worthiness. The worthiness parameter is defined as a discrete choice variable, which compounds the effects of many traffic and object properties. Rather than classifying the objects into two classes based on its cache-ability, this scheme classifies the objects



**Fig. 15** Adaptability check: cache performance under concept drift. Shows that the proposed method is capable of withstanding abrupt and small shifts



into multilevel classes. As this scheme is flexible, it can be fine tuned by adding or removing predictive variables, in order to match a specific form of caching or content distribution.

We used the MLR model to construct the classification scheme. The MLR model is capable of learning from the samples and predict the outcomes. Experiments have shown that the model has a good prediction capability, and is suitable for the adaptive nature of content distribution.

We used a modified version of the LRU algorithm LRU-M to test the effectiveness of the content classification. The LRU-M performance is compared with another class based algorithm (LRU-C). We observe that the cache replacement performs better with the multi class information. The new performance parameters WHR and WBHR reflect the overall performance of the cache rather than only from the users point of view. Thus they reflect the actual effect on the deployed cache.

More research may be useful to determine the exact relationship between the worthiness factor and cache performance. The performance and the cost comparison of the MLR model with the other machine learning techniques such as SVM and Decision Trees is suggested as a topic for further research.

## References

- Agresti A, Wiley J (1990) *Categorical data analysis*, vol 1, 2nd edn. Wiley, New York
- Ahn H, Moon H, Fazzari M, Lim N, Chen J, Kodell R (2007) Classification by ensembles from random partitions of high-dimensional data. *Comput Stat Data Anal* 51(12):6166–6179
- Attar V, Sinha P, Wankhade K (2010) A fast and light classifier for data streams. *Evol Syst* 1(3):199–207. doi:10.1007/s12530-010-9010-1
- Bahn H, Koh K, Noh S, Lyul S (2002) Efficient replacement of nonuniform objects in web caches. *Computer* 35(6):65–73
- Bian N, Chen H (2008) A least grade page replacement algorithm for web cache optimization. In: *Knowledge discovery and data mining*, 2008. WKDD 2008. First international workshop on, pp 469–472
- Breslau L, Cao P, Fan L, Phillips G, Shenker S (1999) Web caching and Zipf-like distributions: evidence and implications. *IEEE INFOCOM* 1(1):126–134
- Cao P, Irani S (2002) Cost-aware www proxy caching algorithms. *IEEE Trans Comput* 51(6):193–206
- Chen X, Zhang X (2003) A popularity-based prediction model for web prefetching (No. 3). IEEE Computer Society Press, Los Alamitos
- Chu F, Wang Y, Zaniolo C (2004) An adaptive learning approach for noisy data streams. In: *Data mining, 2004. ICDM '04. Fourth IEEE international conference on*, pp 351–354
- Cobb J, ElAarag H (2008) Web proxy cache replacement scheme based on back-propagation neural network. *J Syst Softw* 81(9):1539–1558
- Dill S, Kumar R, McCurley K, Rajagopalan S, Sivakumar D, Tomkins A (2002) Self-similarity in the web. *ACM Trans Int Technol* 2(3):205–223
- Dreiseitl S, Ohno-Machado L (2002) Logistic regression and artificial neural network classification models: a methodology review. *J Biomed Inform* 35(5–6):352–359
- Dreiseitl S, Ohno-Machado L, Kittler H, Vinterbo S, Billhardt H, Binder M (2001) A comparison of machine learning methods for the diagnosis of pigmented skin lesions. *J Biomed Inform* 34(1):28–36
- Efron B, Gong G (1983) A leisurely look at the bootstrap, the jackknife, and cross-validation. *Am Stat*, pp 36–48
- Foong AP, Hu Y-H, Heisey DM (1999) Logistic regression in an adaptive web cache. *IEEE Int Comput* 3(5):27–36
- Friedman J, Hastie T, Tibshirani R (2000) Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors). *Ann Stat* 28(2):337–407
- Gama J, Medas P, Castillo G, Rodrigues P (2004) Learning with drift detection. *Lect Notes Comput Sci* 1:286–295
- Gao J, Fan W, Han J, Yu PS (2007) A general framework for mining concept-drifting data streams with skewed distributions. In: *Proceedings of SDM*
- Gonzalez-Canete FJ, Casilari E, Trivino-Cabrera A (2006) Two new metrics to evaluate the performance of a web cache with admission control. In: *Electrotechnical conference, 2006. MEL-ECON 2006. IEEE mediterranean*, pp 696–699
- Green M, Björk J, Forberg J, Ekelund U, Edenbrandt L, Ohlsson M (2006) Comparison between neural networks and multiple logistic regression to predict acute coronary syndrome in the emergency room. (No. 3). Tecklenburg, Federal Republic of Germany, Burgverlag, c1989
- Hosmer D, Lemeshow S (2000) *Applied logistic regression*, vol 354, 2nd edn. Wiley, New York. <http://books.google.com/books?id=Po0RLQ7USIMC>

- Imai K, King G, Lau O (2006) Zelig: everyone's statistical software. <http://gking.harvard.edu/zelig>
- Jin S, Bestavros A (2000) Popularity-aware greedy dual-size web proxy caching algorithms. In: Distributed computing systems, 2000. Proceedings. 20th international conference on, pp 254–261
- Klinkenberg R, Renz I (1998) Adaptive information filtering: learning in the presence of concept drifts. *Learn Text Categor* 1:33–40
- Komarek P (2004) Logistic regression for data mining and high-dimensional classification. *Biostatistics* 4:138
- Koskela T, Heikkonen J, Kaski K (2003) Web cache optimization with nonlinear model using object features. *Comput Netw* 43(6): 805–817
- Krashakov SA, Teslyuk AB, Shchur LN (2006) On the universality of rank distributions of website popularity. *Comput Netw* 50(11): 1769–1780
- Krashakov SA, Teslyuk AB, Shchur LN (2006) On the universality of rank distributions of website popularity. *Comput Netw* 50(11): 1769–1780
- Landwehr N, Hall M, Frank E (2005) Logistic model trees. *Mach Learn* 59(1):161–205
- Li K, Nanya T, Qu W (2007) A minimal access cost-based multimedia object replacement algorithm. In: IEEE international parallel and distributed processing symposium, 2007. IPDPS 2007, pp 1–7
- Lim T, Loh W, Shih Y (2000) A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Mach Learn* 40(3):203–228
- Long W, Griffith J, Selker H, D'agostino R (1993) A comparison of logistic regression to decision-tree induction in a medical domain. *Comput Biomed Res* 26:74–97
- Lu Y, Abdelzahr T, Lu C, Tao G (2002) An adaptive control framework for QoS guarantees and its application to differentiated caching. In: Quality of service, 2002. Tenth IEEE International Workshop on, pp 23–32
- Markatchev N and Williamson C (2002) Webtraff: A GUI for web proxy cache workload modeling and analysis. In: Modeling, analysis and simulation of computer and telecommunications systems, 2002. MASCOTS 2002. Proceedings. 10th IEEE international symposium on, p 356–363
- Mierswa I, Wurst M, Klinkenberg R, Scholz M, Euler T (2006) Yale: rapid prototyping for complex data mining tasks. In: Ungar L, Craven M, Gunopulos D, Eliassi-Rad T (eds) *Kdd '06: Proceedings of the 12th acm sigkdd international conference on knowledge discovery and data mining*. ACM Press, New York, NY, USA, pp 935–940
- Miller A (2002) *Subset selection in regression*. CRC Press, New York
- NLANR (2010) Cache access logs [online]. <ftp://ircache.nlanr.net/traces/>
- Pallis G, Thomos C, Stamos K, Vakali A, Andreadis G (2007) Content classification for caching under CDNs. In: Innovations in information technology, 2007. IIT '07. 4th international conference on, pp 586–590
- Podlipnig S, Böszörmenyi L (2003) A survey of web cache replacement strategies. *ACM Comput Surv* 35(4):374–398
- Sargent D (2001) Comparison of artificial neural networks with other statistical approaches. *CA A Cancer J Clin* 91(S8):1636–1642
- Steyerberg EW, Harrell FE, Borsboom GJJM, Eijkemans MJC, Vergouwe Y, Habbema JDF (2001) Internal validation of predictive models: efficiency of some procedures for logistic regression analysis. *J Clin Epidemiol* 54(8):774–781
- Sulaiman S, Shamsuddin SM, Forkan F, Abraham A (2008) Intelligent web caching using neurocomputing and particle swarm optimization algorithm. In: *Ams '08: Proceedings of the 2008 second asia international conference on modelling & simulation (ams)*. IEEE Computer Society, Washington, DC, pp 642–647
- Team RDC (2008) R: a language and environment for statistical computing. R Language software Team, Vienna
- Tian W, Choi B, Phoha VV (2002) An adaptive web cache access predictor using neural network. In: *Iea/aie '02: Proceedings of the 15th international conference on industrial and engineering applications of artificial intelligence and expert systems*. Springer, London, pp 450–459
- TraceGraph (2005) Trace graph tool (online). <http://www.tracegraph.com/traceconverter.html>
- Tsymbol A (2004) The problem of concept drift: definitions and related work. Computer Science Department, Trinity College, Dublin
- Wang H, Fan W, Yu PS, Han J (2003) Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining, pp 226–235
- Wang Y (2005) A multinomial logistic regression modeling approach for anomaly intrusion detection. *Comput Secur* 24(8):662–674
- Xu L, Chow M-C, Gao XZ (2005) Comparisons of logistic regression and artificial neural network on power distribution systems fault cause identification. In: *Soft computing in industrial applications, 2005. SMCia/05. Proceedings of the 2005 IEEE Midsummer workshop on*, pp 128–131
- Yang Q, Zhang HH (2003) Web-log mining for predictive web caching. *IEEE Trans Knowl Data Eng* 15(4):1050–1053