Harnessing Hundreds of Millions of Cases: Case-Based Prediction at Industrial Scale

Vahid Jalali[1] and David Leake[2]

[1] Walmartlabs
Sunnyvale CA 94086, USA
vjalali@walmartlabs.com
[2] School of Informatics, Computing, and Engineering
Indiana University, Bloomington IN 47408, USA
leake@indiana.edu

Abstract. Building predictive models is central to many big data applications. However, model building is computationally costly at scale. An appealing alternative is bypassing model building by applying case-based prediction to reason directly from data. However, to our knowledge case-based prediction still has not been applied at true industrial scale. In previous work we introduced a knowledge-light/data intensive approach to case-based prediction, using ensembles of automatically-generated adaptations. We developed foundational scaleup methods, using Locality Sensitive Hashing (LSH) for fast approximate nearest neighbor retrieval of both cases and adaptation rules, and tested them for millions of cases. This paper presents research on extending these methods to address the practical challenges raised by case bases of hundreds of millions of cases for a real world industrial e-commerce application. Handling this application required addressing how to keep LSH practical for skewed data; the resulting efficiency gains in turn enabled applying an adaptation generation strategy that previously was computationally infeasible. Experimental results show that our CBR approach achieves accuracy comparable to or better than state of the art machine learning methods commonly applied, while avoiding their model-building cost. This supports the opportunity to harness CBR for industrial scale prediction.

Key words: Big Data, Case Based Reasoning, Ensemble of Adaptations, Locality Sensitive Hashing, Skewed Data

1 Introduction

Predicting future customer actions is integral to e-commerce competitiveness. For example, the success of retailers is closely tied to predicting customer behaviors to maximize effectiveness of the supply chain, inventory management, and marketing. The standard method for such prediction is to abstract data into mathematical models to apply to the prediction task (e.g., a firm might train a logistic regression model to predict whether a customer will make an electronics purchase within the next month). The staggering growth of digital data has

made a plethora of training instances available, increasing prediction opportunities but making model building computationally challenging. This challenge is commonly addressed at two stages: First, by applying sampling methods to select a manageable-size subset of the data to use as a training set (e.g., [26]), and second, by applying optimization techniques such as stochastic learning (e.g., [4]) or parallelization (e.g., [31]) to expedite the model-building process.

In contrast to model-based approaches, case-based reasoning's lazy learning approach skips the model building step entirely, to retrieve cases on demand and predict directly from them. This avoids the cost of model building but shifts cost to retrieval. Often retrieval costs are not a practical impediment [9]. However, successful industrial-scale CBR will depend on controlling them. A rich current of CBR research develops methods for controlling retrieval costs by compressing the case base (see Leake, Smyth, Wilson and Yang [18] for some examples). However, applying even carefully-crafted competence-driven methods (e.g., [30]) has two major drawbacks. First, deletion of cases may unavoidably lose information. Second, compression can impose a considerable pre-processing cost penalty [9]. Standard compression methods generally have $O(n^2)$ complexity [30], making them prohibitively expensive to apply to case bases at scale.

In previous work [11] we argued that using retrieval methods from big data frameworks can enable scaling up CBR without compression. That work introduced large scale case/rule retrieval and adaptation generation methods for regression and classification tasks, using Locality Sensitive Hashing (LSH) [10] for efficient approximate nearest neighbor retrieval, based on examining a subset of candidate cases (those in the the same hash bucket as the query) rather than the entire case base. Evaluations supported that the methods were practical for large case bases (tests included a case base of two million cases), but also revealed that handling larger case bases would require going beyond those methods alone. It is now common to deal with data sets with several hundreds of millions of instances [1], and is not unheard of to handle billions of instances [19, 23]. Consequently, new methods are needed.

This paper discusses the primary challenges of extending the LSH-based case-based prediction approach to the next level, of hundreds of millions of cases, and how we addressed them. The testbed domain for this work is an important e-commerce application: predicting the propensity of the users of an e-commerce platform to engage with a marketing vehicle (e.g., to open a marketing email) or to make a transaction within a certain set of departments over a certain period of time in future. These tasks require predicting probabilities ranging from 0 to 1. The data in this domain is skewed and the labels are imbalanced. Skewed data is a common problem in real-world domains and has been the subject of considerable study in machine learning (e.g., [5, 3]), but has received only limited consideration in CBR (e.g., [21, 20]).

This paper proposes and evaluates a set of case-based predictors able to retrieve sufficiently efficiently to handle case bases two orders of magnitude larger than previously tested and to deal with skewed data. It proposes three new methods for efficient large-scale retrieval by LSH when handling skewed data,

hierarchical LSH, prototype-based case retrieval, and capping the number of cases within a hash bucket. It presents an CBR approach applying these methods, *Ensembles of Adaptations at ScalE* (EASE), building from our previous work using automatically generated ensembles of adaptations [11–13, 15], and tests an implementation for a sampling of variant methods.

The rest of the paper is organized as follows. First, it reviews existing work on large scale case-based reasoning. Next, it introduces EASE. It then presents an evaluation of four different configurations of EASE, to identify the contributions of different design choices, as well as evaluation of a new configuration of our ensemble-based adaptation learning/application method, enabled by the new retrieval methods, exploiting much larger sets of automatically-generated adaptation rule ensembles than in our previous work. The paper closes with a summary of contributions and future directions.

2 Background: Retrieval Cost and Case-Base Growth

2.1 Speedup Through Maintenance

The CBR community has long been aware that case base growth can impair system efficiency, due to the utility problem as retrieval cost grows due to increased case base size (e.g., [28]). This has led to much research on case-base maintenance methods for controlling case base size, such as selective retention of cases (e.g., [24, 27]) and competence-based case deletion/selection (e.g., [29, 30, 32]). Retention strategies have also been developed to control growth of automatically-generated case adaptation rule sets [16].

However, even state of the art competence-based techniques cannot ensure protection against knowledge loss from deleted cases or rules. As compression increases competence loss can become severe (e.g., [30]). In addition, sophisticated competence-preserving deletion methods are computationally expensive—overwhelmingly so for data at scale—and must be incurred each time the case base is compressed [9]—which may occur routinely in the life cycle of a CBR system. These issues with compression methods motivate replacing compression by retrieval efficient enough to handle complete case bases at scale.

2.2 Speedup by Big Data Methods

Recent work on applying big data methods to CBR provides a first step towards large-scale retrieval without knowledge loss, but has limitations. Some existing methods enable rapid retrieval of exact matches, enabling efficiency and accuracy to be achieved simultaneously when similarity-based matching can be sacrificed. For example, Dumoulin [2] applied a MapReduce based retrieval method to perform efficient exact match retrieval on a case base of several million cases.

However, CBR normally requires searching for nearest neighbors of a case, for partially-matching neighbors. In these scenarios, locality-sensitive hashing (LSH) is a natural candidate for practical approximate nearest neighbor search.

Locality Sensitive Hashing: Locality Sensitive Hashing (LSH) [10] is an efficient method for mapping data points into "buckets." With appropriate parameter settings LSH is likely to place similar points in the same bucket. LSH is often used to speed up nearest neighbor search by narrowing down the search space for a given query to a subset of cases—those in the bucket associated with that query—rather than the whole case base. LSH is an approximation method; it does not guarantee that all nearest neighbors of a case will be grouped into the same bucket nor does it guarantee that all cases in the same bucket will be similar to each other. However, LSH has been shown sufficiently accurate to be an effective practical approach for finding nearest neighbors of a case [6], and the trade-off between accuracy and efficiency can be tuned as needed (e.g. [17]). Various schemes have been developed to improve its core method [6, 7, 17].

Integrating LSH with CBR: We began to explore LSH for similarity-based retrieval in our work on BEAR [11], which applied LSH using p-stable hashing [6] for retrieval for case-based numerical prediction (regression) tasks. BEAR was tested on case bases with up to two million cases. We recently introduced EACH [12], a locality sensitive hashing scheme for domains with both categorical and numeric input features/target values, and used it as the basis of fast large-scale retrieval in EACX, a scalable case-based classifier. This work showed that applying ensembles of adaptations to adjust the approximate nearest neighbor solutions can help compensate for the lower retrieval quality of LSH retrieval compared to full nearest neighbor retrieval [11, 12].

3 Ensembles of Adaptations at Scale (EASE)

3.1 Foundations of EASE

The EASE method builds on succession of knowledge-light approaches for improving CBR performance by automatically generating adaptation rules from the case base and applying ensembles of those solutions for problem-solving and classification [11–13, 15]. Experimental results support that these methods significantly increase accuracy over baselines. Here we summarize the two most relevant variants for the prediction tasks of this paper: EAR (Ensembles of Adaptation for Regression) [15], which developed the basic, approach, and BEAR (Big Data Ensembles of Adaptations for Regression) [11], which scaled up the approach. However, the scaleup methods from this paper could be applied to the classification variants as well.

EAR uses the *Case Difference Heuristic* (CDH) [8] approach to generate adaptations from pairs of cases in the case base. Each resulting rule has two parts, a description of differences in the input features of the two cases and a description of the differences between their solutions (here, values or labels). EAR generates adaptations by comparing pairs of cases, selected by one of three strategies:

1. Local cases - Local neighbors: Generate adaptations by comparing every pair of cases in the local neighborhood of the input query. Because this approach

considers few cases, it is practical for lazy generation of adaptation rules on demand.
2. Global cases - Local neighbors: Generate adaptations by comparing every case in the case base with its few top nearest neighbors.
3. Global cases - Global cases: Generate adaptations by comparing every possible pair of cases in the case base.

EAR solves input problems by retrieving a relevant prior case or cases, according to a selection strategy, and applying ensembles of automatically-generated adaptations to generate sets of solutions to combine.

BEAR scales up EAR by using LSH for fast approximate nearest neighbor retrieval. For any retrieval task, the appropriate LSH method is dictated by the input features' data types in the underlying domain; BEAR used p-stable hashing. BEAR developed foundations for case-based predictors at scale, handling millions of cases. However, it did not address the scale and skewed data issues of our e-commerce domain.

3.2 EASE

EASE extends BEAR to handle arbitrarily large case bases, where the limitation is only the storage capacity of the underlying big data platform—*the expected look up time remains constant as the size scales up, even for skewed data.* To reduce computation costs, EASE dedupes cases with identical input features and uses collision handling methods to handle cases with non-identical but similar features.

Case/Adaptation Deduping: In our e-commerce domain, cases often have identical input features. For example, when predicting customer propensity to engage with marketing emails, based on cases for past customer actions, it is likely that many customers never opened or clicked a marketing email, yielding identical features (here, zero opened or clicked marketing emails over any of a range of time periods). EASE handles such cases through a deduping process, separate from the skewed data process that applies to distinct cases/adaptations whose similar input features result in their being hashed to the same hash bucket. Deduping cases with identical input features serves two purposes. First, deduping improves efficiency. Cases with identical input features in the training data have the same performance effect on LSH as skewed data, because excessive numbers of cases in the same hash bucket make nearest neighbor retrieval inefficient within the hash bucket. Second, duplicate cases can degrade the accuracy of prediction if there is high variation in their labels. For example, if there are millions of such cases with numeric target values with very high variance, randomly selecting a few of these cases as the nearest neighbors of the input query will result in high variance for predictions. In addition, if all nearest neighbors of an input query have identical input features, then EASE's automatic adaptation generation would generate the same difference vector for comparing the input

query with its nearest neighbors, which in turn would reduce diversity in the adaptation rules to be applied and decrease the benefit of ensemble adaptation.

Deduping in EASE is done at both training and solution building stages and is applied to both cases and adaptation rules. At training, cases/adaptations with identical features are grouped and form a prototype case whose solution is a function of all participating cases' solutions (e.g. the mean or median). Techniques such as outlier removal could be used to generate a better estimate of the prototype case's solution in presence of noise.

Deduping during solution-building is done differently depending on whether EASE is run in batch mode or to process streaming problems. In batch mode all input queries are known to the system in advance. In the streaming mode, input queries are introduced to the system successively and not known to the system in advance. At the solution building stage, in the batch mode, only one version of cases with identical features is retained. The calculated solution is then replicated for all cases with identical input features. In the streaming mode, the solutions for previously solved problems are stored and each new incoming problem is looked up in the pool of previously solved problems.

Collision Handling: The main novelty of EASE's approach is its collision handling module. The collision handling module controls the number of cases per hashing bucket with the aim of keeping the number of cases per bucket within a desired range to ensure efficient case/rule retrieval. We have identified two families of methods for collision handling for EASE, one lossy and one lossless. To the best of our knowledge, neither has been proposed previously.

Lossy Collision Handler: The lossy handler deletes cases in a bucket to keep the number of cases below a threshold. This approach may seem incompatible with EASE's claim of avoiding the information loss associated with case-based compression. However, case deletion in EASE is mainly targeting skewedness in the data, with the premise that given many cases with similar input features, keeping all those cases does not improve prediction accuracy. We propose two types of lossy collision handlers, competence-based and sampling-based:

- Competence-based Collision Handler: Uses competence-based deletion methods to control the number of cases per bucket. For example, a footprint deletion policy [29], or—more relevant to the ensemble and adaptation-based nature of EASE—adaptation-guided maintenance [16] could be applied.
- Sampling-based Collision Handler: Uses sampling methods to control the number of cases per bucket. The most naive such method is random sampling. However, more advanced sampling techniques such as clustering or density-sensitive sampling [25] could also be used. Random sampling has the advantage of low time complexity compared to advanced sampling techniques or competence-based alternatives. We note that an overly expensive collision handling step could easily become a new bottleneck, nullifying speed gains from EASE.

Lossless Collision Handler: Another approach to deal with a high collision rate is to keep all cases, but to further split the cases into new buckets to guarantee the limit on the maximum bucket size. LSH controls the number of cases in each bucket thorough the *sensitivity* of the chosen hashing functions. Increasing the sensitivity increases the total number of buckets while decreasing the average number of cases per bucket; decreasing the sensitivity reduces the number of buckets and increases the average number of cases per bucket. For large case bases with skewed data distributions, two conflicting factors pose a problem for LSH. Increasing sensitivity results in a large number of buckets with few cases per bucket, potentially making it unlikely there will be enough candidate cases/rules for ensemble-based solution building by EASE. On the other hand, decreasing sensitivity results in a manageable number of buckets, but a few of these buckets will still contain a large number of cases. For cases hashed to those buckets the retrieval process may have excessive computational cost. We address this with a method we call *Hierarchical LSH*.

Hierarchical LSH: The main idea of *Hierarchical LSH* is to use a moderate sensitivity to hash cases initially, and then to apply additional rounds of hashing with higher sensitivity to buckets with high collision level (i.e. buckets with large number of cases). This results in a density sensitive LSH that tunes the sensitivity for different subspaces according to their case distribution densities. We note that adding more levels of hashing can potentially increase the look up time, but practically speaking the time complexity should be comparable with one level LSH. If there are i cases in a skewed hash bucket and if r, is the number of cases per bucket considered manageable, then the number of required hash levels will be $O(\log_r i)$.

These collision handling methods apply to both batch and streaming scenarios. In streaming scenarios, reservoir sampling can be used for random sampling. Both footprint and adaptation-guided collision handling apply directly to streaming settings. Because hierarchical LSH keeps all cases in the case base, it requires no special treatment for streaming scenarios. However, because of cost of competence-based maintenance, random sampling and hierarchical LSH are the most promising collision-handling methods for very large case bases. The following experiments test EASE with random sampling.

3.3 Adaptation Generation and Building the Solution

EASE uses two adaptation generation alternatives: Local cases - Local neighbors (henceforth local-local), and Global cases - Local neighbors (henceforth global-local). Because the extreme large size of the case base, Global cases - Global cases would impose overwhelming computational costs. In terms of space complexity, Local cases - Local neighbors requires constant storage, and Global cases - Local neighbors requires storage of the order of the case base size.

Algorithm 1 summarizes EASE's solution building process. First, the input query is hashed, using the chosen hashing method, and assigned to a case bucket. EASE then does nearest neighbor search among the cases in the same bucket as

Algorithm 1 EASE's basic algorithm

Input:
Q: query
n: number of source cases to adapt to solve query
r: number of adaptation rules to apply per source case
CB: case base
sample: whether or not to sample the training data
mode: rule generation mode (local-local or global-local)

Output: Estimated solution value for Q

```
//Begin Preprocessing .....
if sample then
   CB ← StratifiedSampler(CB)
end if
HashedCaseBase ← LSH(CB)
if mode == global-local then
   rules ← RuleGenerator(HashedCaseBase)
   HashedRules ← LSH(rules)
end if
//End Preprocessing .....
if mode == local-local then
   rules ← RuleGenerator(HashedCaseBase, Q)
   HashedRules ← LSH(rules)
end if
CasesToAdapt ← ApproximateNeighborhoodSelector(Q,n,HashedCaseBase)
for c in CasesToAdapt do
      RulesToApply ← ApproximateNeighborhoodSelector(HashedRules,c,Q)
      SolutionEstimate(c) ← MajorityRuleVote(RulesToApply, c, r)
end for
return MajorityVote(∪_{c∈CasesToAdapt} SolutionEstimate(c))
```

the input query. By comparing the features of the input query and its nearest neighbors (here the source cases), EASE generates a set of difference descriptors. These difference descriptors are hashed and assigned to the corresponding adaptation buckets. Nearest neighbor search is done within each adaptation bucket and the top K nearest adaptation rules (for a pre-set K) are retrieved for each difference descriptor. For each source case, the new values proposed by the adaptations are aggregated and used to adjust the source case values. The adjusted values for each source case are combined to form the final solution.

3.4 EASE Architecture

Figure 1 presents the EASE architecture, for the configuration in which adaptations are generated by comparing each case with its top nearest neighbors (global-local). The architecture for other variations of EASE is very similar, with slightly simpler flow.

As a preprocessing step, cases in the case base are hashed into different buckets. The input query is hashed using the same mechanism to identify the relevant bucket for the query. The "approximate nearest neighbor retriever" selects its approximate nearest neighbors, whose solutions are adjusted and used to build the final solution. The collision handler module ensures that the number of cases within each hash bucket does not exceed a pre-set threshold, to guarantee acceptable run time. Adaptations are generated by comparing cases in the same hash bucket and form the rule base.

As described in detail in Jalali and Leake [15, 13], the EAR/EAC approach automatically generates adaptation rules from cases, indexed analogously to cases; this enables applying the same retrieval process to cases and rules. The rule base is partitioned into different buckets using LSH, and as for the hashed case base, the collision handler ensures that the number of adaptations per bucket does not exceed a certain limit. To generate adaptation rules, the input query and source cases retrieved by the "approximate nearest neighbor retriever" query are compared and their feature differences are generated and hashed. To generate a solution for an input query, the solution of each source case is adjusted by applying an ensemble of adaptations addressing problems with differences similar to those between the input query and that source case. The final solution is built by combining the adjusted values of the retrieved source cases.

4 Evaluation

Our evaluations test the accuracy and efficiency of different variants of EASE and compare them to two classic machine learning approaches for industrial big data, Logistic Regression and Random Forest. Specifically, the evaluations address the following questions:

1. Comparative accuracy: How does the accuracy of different variants of EASE compare to each other and baselines?
2. Effect of preprocessing sampling on performance: How sensitive is the accuracy of EASE to using the whole training data without any preprocessing compared to preprocessing the data with stratified sampling?
3. Comparative execution time: How does the execution time of EASE compare to baseline methods?

4.1 Experimental Design

We tested EASE for two data sets from real-world problems. The first is the Kaggle Real Time Bidding data set[3], which is publicly available and contains one million cases. Kaggle data sets are posted as practical challenges for predictive modeling and analytics competitions. The second is data for building propensity models for Walmart.com, an e-commerce platform with hundreds of millions of users. Because this data contains several hundred million cases, it far exceeds

[3] https://www.kaggle.com/zurfer/rtb/data

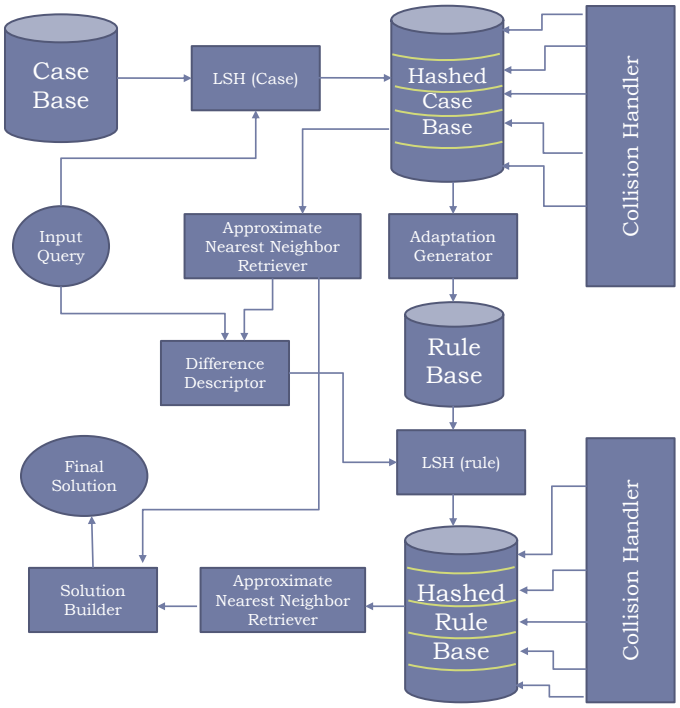

Fig. 1. EASE architecture

the scope of previous CBR applications. We respect corporate practice by not providing precise proprietary details such as the exact size or features. However, because we report the relative performance of the tested methods compared to baselines applied to the same data, comparative performance can be assessed, and these relative results can be compared to the fully replicable results on the Real Time Bidding data set.

Evaluation Tasks: The first task is to predict whether a user will open a marketing email within seven days from receiving it. We will refer to this task as engagement propensity. The input features to predict engagement propensity are mainly related to users' previous interactions with marketing email, such as how many emails they have historically received, opened or clicked. Each user case includes tens of such features. The second task is to predict whether a user will make a purchase within the health and beauty departments in the next 30 days. We refer to this task as purchase propensity. The input features to predict purchase propensity are mainly related to user past purchases and browsing behavior. Each user case includes hundreds of such features. The two tasks are specifically selected to study EASE's performance under different ratios of label values where the class imbalance issue is more severe for propensity prediction. The third task is to predict whether a user will click on an ad (e.g. banner on a

webpage). We refer to this task as click prediction. There are 88 input features in the click prediction domain. The data is anonymized by Kaggle; PCA was applied to the original features and 0.99 of the linear explanatory power was retained. The original features included browser, operating system or time of the day the user is online, etc. There are 1908 cases with label 1 and the rest of the cases in this domain have 0 as their labels.

Evaluation Methodology: For engagement and purchase propensity models we pick training labels from one week and one month of user activities respectively and evaluate the models on the activities of the users over the following week and month. For click prediction we randomly split the data to 70% training and 30% test. Because the tests deal with class imbalance, we use the area under the Precision-Recall (PR) curve to evaluate different methods rather than accuracy, AUC or other metrics. The PR curve is especially informative if propensity predictions are used to pick customers to receive a marketing campaign where the objective is to maximize the number of customers that convert (open an email or make a purchase in a specific set of departments). In this case the PR curve can give us the expected number of customers that convert at different recall levels which can be translated to segment sizes (i.e. number of recipients of the marketing campaign).

Implementation: We implemented all EASE variations in Apache Spark and used BucketedRandomProjectionLSH class from Apache Spark MLlib [22] for LSH. We used 0.25 as the sensitivity of the LSH across all variations of EASE. This was selected experimentally based on the bucket size distributions it yields for the e-commerce data. We set the number of hash tables in BucketedRandomProjectionLSH to six; i.e., every case/rule is hashed using six hash functions. These two parameters (i.e. sensitivity and number of hash tables) could be treated as hyper parameters, but in order to narrow down the search space (for hyper parameter tuning) we picked their values experimentally. Spark MLlib provides built-in functionality for grid search, which we used for hyper parameter tuning. It also contains classic predictors such as Logistic Regression (LR) and Random Forest (RF) which we have used in our evaluations. The hyper parameters we tuned for LR are the maximum number of iterations and the regularization parameter; for RF they are the number of the tress and their maximum depth. In our evaluations we used entropy as the underlying impurity measure in RF. The area under the PR curve is calculated using the BinaryClassificationMetrics class from Spark MLlib. We used random sampling within buckets as the underlying collision handling method in our implementation of EASE. We implemented and tested four variants of EASE:

1. EASE-Strat-Local: Uses LSH for approximate nearest neighbor search for case and rule retrieval. It uses the stratified version of the case base as the training set and generates adaptations from the local neighborhood of the input query only. The hyper parameters of EASE-Strat-Local are: number of approximate nearest neighbors to use, number of adaptations to apply, and maximum number of cases per bucket.

Table 1. Gain in area under PR curve of EASE-Strat-Local, EASE-Full-Local, EASE-Strat-Global, EASE-Full-Global, LR and RF over the baseline for engagement and purchase propensity prediction tasks

Task name	**EASE Strat-Local**	**EASE Full-Local**	**EASE Strat-Global**	**EASE Full-Global**	**LR**	**RF**
Engagement Propensity	8.02%	7.42%	5.92%	5.17%	10.30%	**12.02%**
Purchase Propensity	19.13%	29.01%	49.94%	**65.41%**	20.58%	11.54%
Click Prediction	20.68%	39.72%	118.39%	**145.86%**	83.32%	72.30%

2. EASE-Full-Local: Differs from EASE-Strat-Local in using the whole training set rather than the stratified version of the training data.
3. EASE-Strat-Global: Differs from EASE-Strat-Local in generating adaptations by comparing each case with its approximate nearest neighbors, rather than from the local neighborhood of the input query. In addition to the hyper parameters listed for EASE-Strat-Local, EASE-Strat-Global requires tuning the number of approximate nearest neighbors of cases to be used in adaptation learning.
4. EASE-Full-Global: Differs from EASE-Strat-Global in using the whole training set rather than the stratified version of the training data.

In addition to these variants, we used an extreme approximate predictor as the baseline. This baseline uses LSH to hash cases in the training set and creates prototype cases by averaging the input features of all cases in the same hash bucket and averaging of their labels. The only hyper parameter for this baseline is the number of nearest prototypes to use building the solution.

4.2 Experimental Results

Questions 1 and 2: Comparative Accuracy and Effect of Preprocessing: We conducted experiments to evaluate the accuracy (in terms of area under PR curve) of different variations of EASE, LR, and RF compared to the baseline method. Table 1 shows the gain in area of these methods over the baseline.

The experiments show the superior performance of EASE compared to other methods under class imbalance settings. For purchase propensity, EASE-Full-Global shows 65% improvement over the baseline method while LR and RF only show 21% and 11% improvement over baseline respectively. For click prediction EASE-Full-Global shows 146% improvement over the baseline while LR and RF show 83.32% and 72.30% improvement respectively. The absolute values

of area under PR curve for EASE-Strat-Local, EASE-Full-Local, EASE-Strat-Global, EASE-Full-Global, LR and RF is 0.0027, 0.0031, 0.0049, 0.0055, 0.0041, and 0.0039 respectively. We hypothesize that the superior performance of EASE using global-local adaptation generation (i.e. EASE-Strat-Global and EASE-Full-Global) over EASE with local-local adaptation generation (i.e. EASE-Strat-Local and EASE-Full-Local) arises because, when there are very few instances with a certain label values in the case base (i.e. class imbalance), it becomes less likely to have enough of these instances in the local neighborhood of the input query and therefore, the generated adaptations will be more homogenous. However, with global-local adaptation generation there will be more diversity in the generated adaptations, providing more benefit from ensemble adaptation. We hypothesize that the superior performance of EASE over model-based machine learning methods such as LR and RF for class imbalance follows from its lazy nature, which enables considering the whole case base rather than abstracting a portion into the mathematical models.

When class labels are more evenly distributed (i.e. engagement propensity setting), the gap between EASE performance compared to classic machine learning models such as LR and RF is smaller and in fact LR and RF slightly outperform EASE but the difference is not drastic (compared to EASE-Strat-Local, 2% for LR and 4% for RF). Among EASE variants, using the local-local method for generating adaptations yields relatively better performance, as shown in Table 1. We hypothesize that sufficient learning opportunities in vicinity of the input query and lack of consideration of context in the global-local method (cf. [14]) are the main reasons for local-local method's superior performance in this case.

The experiments also show that for engagement propensity prediction, there is only a modest difference between using the stratified version versus using the full the training data, with the stratified version showing relatively better performance. In the purchase propensity task, the strong class imbalance setting, the non-stratified versions of EASE show better performance and this gap is especially marked when the global-local method is used for generating the adaptations. Overall, we conclude that pre-processing the case base with stratified sampling is not necessary for EASE. This can save time and computational resources, and is made feasible in practice by the scalability of EASE.

Question 3: Execution Time: The execution time of different methods depends on the underlying technology used to implement them. We implemented EASE in spark to be able to code everything ranging from LSH, to grid search, to evaluation metrics in the same environment. Benchmarking the performance of EASE implemented in Spark versus in other technologies such as NoSQL databases is out of the scope of this work; we only report results based on Spark execution times. We ran Spark on an Apache Mesos cluster with hundreds of workers, several thousands of CPU units and Terabytes of memory. Preprocessing the case base of several hundreds of millions of cases and conducting stratified sampling takes tens of minutes processing time and building a single model (characterizing a single set of departments, e.g., health and beauty) usually takes a few minutes. Scaling these up to a scenario in which more models are required for

different super department sets or engagement with different marketing vehicles, the execution time for the model-based approach can go beyond several hours. Using EASE makes it possible to avoid the preprocessing and model building steps and to save this time and computational resources.

Building the solution for an input query usually takes several to several hundreds of milliseconds for model based approaches such as LR or RF depending on the number of features and the complexity of the model. This value increases to a few seconds for EASE's implementation in Spark. However, we believe by using NoSQL databases the run time of EASE can be reduced to a few lookups and a few mathematical operations such as calculating the distance between cases, picking the top nearest neighbors and taking the average of a few values. Considering that a lookup only costs a few milliseconds in NoSQL databases, we believe that the whole solution building process can be kept below a second using the appropriate storage solution. We expect the result to be that EASE will provide much shorter preprocessing time and comparable solution building time to model-building methods.

5 Conclusion and Future Work

This paper introduced Ensemble of Adaptations at ScaleE (EASE), a case-base predictor for industrial big data settings that can handle skewed data. A central goal of this project is to scale up CBR to handle case base sizes far beyond those of previous studies and make CBR methods competetive for large-scale big data prediction tasks.

EASE uses locality sensitive hashing to perform approximate nearest neighbor search. LSH nearest neighbor sacrifices some accuracy for the sake of efficiency. However, the ensemble-based nature of EASE is able to compensate for its accuracy loss. EASE uses deduping and collision handling to maintain efficiency for retrieval from arbitrarily large case bases. We evaluated EASE both for a public real-time bidding domain with one million cases used for a Kaggle challenge, and for building propensity models for an e-commerce platform with hundreds of millions of customers. Experimental results showed superior performance for EASE compared to sample classic machine learning models under class imbalance settings and showed comparable performances under more even label value distribution scenarios. However, compared to other model-based predictors EASE saves time and computational resources by skipping the model building and training data preprocessing step.

The future directions for this work include more extensive performance benchmarking under different levels of class imbalance and for implementations using different platforms such as Apache Spark, NoSQL or an index-based solution such as Elasticsearch. Another direction is adding contextual considerations to adaptation retrieval when adaptations are generated from the entire case base [14]. Yet another is to study the efficiency and performance effects of our alternative proposed collision handling mechanisms such as hierarchical LSH.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: Dbpedia: A nucleus for a web of open data. In: Proceedings of the 6th International The Semantic Web and 2Nd Asian Conference on Asian Semantic Web Conference. pp. 722–735. ISWC'07/ASWC'07, Springer, Berlin (2007)
2. Beaver, I., Dumoulin, J.: Applying mapreduce to learning user preferences in near real-time. In: Case-Based Reasoning Research and Development, ICCBR 2014. pp. 15–28. Springer, Berlin (2014)
3. Bi, Z., Faloutsos, C., Korn, F.: The "dgx" distribution for mining massive, skewed data. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 17–26. KDD '01, ACM, New York (2001)
4. Bottou, L.: Large-scale machine learning with stochastic gradient descent. In: Lechevallier, Y., Saporta, G. (eds.) Proceedings of COMPSTAT'2010. pp. 177–186. Physica-Verlag HD, Heidelberg (2010)
5. Chawla, N.V.: Data mining for imbalanced datasets: An overview. In: Maimon, O., Rokach, L. (eds.) Data Mining and Knowledge Discovery Handbook, pp. 875–886. Springer US, Boston, MA (2010)
6. Datar, M., Immorlica, N., Indyk, P., Mirrokni, V.S.: Locality-sensitive hashing scheme based on p-stable distributions. In: Proceedings of the Twentieth Annual Symposium on Computational Geometry. pp. 253–262. SCG '04, ACM, New York (2004)
7. Gionis, A., Indyk, P., Motwani, R., et al.: Similarity search in high dimensions via hashing. In: VLDB. vol. 99, pp. 518–529 (1999)
8. Hanney, K., Keane, M.: Learning adaptation rules from a case-base. In: Proceedings of the Third European Workshop on Case-Based Reasoning. pp. 179–192. Springer, Berlin (1996)
9. Houeland, G., Aamodt, A.: The utility problem for lazy learners - towards a non-eager approach. In: Bichindaritz, I., Montani, S. (eds.) Case-Based Reasoning Research and Development, ICCBR 2010. pp. 141–155. Springer, Berlin (2010)
10. Indyk, P., Motwani, R.: Approximate nearest neighbors: Towards removing the curse of dimensionality. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing. pp. 604–613. STOC '98, ACM, New York (1998)
11. Jalali, V., Leake, D.: CBR meets big data: A case study of large-scale adaptation rule generation. In: Case-Based Reasoning Research and Development, ICCBR 2015. pp. 181–196. Springer, Berlin (2015)
12. Jalali, V., Leake, D.: Scaling up ensemble of adaptations for classification by approximate nearest neighbor retrieval. In: Case-Based Reasoning Research and Development, ICCBR 2017. pp. 154–169. Springer, Berlin (2017)
13. Jalali, V., Leake, D., Forouzandehmehr, N.: Ensemble of adaptations for classification: Learning adaptation rules for categorical features. In: Case-Based Reasoning Research and Development, ICCBR 2016. pp. 186–202. Springer, Berlin (2016)
14. Jalali, V., Leake, D.: A context-aware approach to selecting adaptations for case-based reasoning. In: Modeling and Using Context, pp. 101–114. Springer, Berlin (2013)
15. Jalali, V., Leake, D.: Extending case adaptation with automatically-generated ensembles of adaptation rules. In: Case-Based Reasoning Research and Development, ICCBR 2013. pp. 188–202. Springer, Berlin (2013)

16. Jalali, V., Leake, D.: Adaptation-guided case base maintenance. In: Proceedings of the Twenty-Eighth Conference on Artificial Intelligence. pp. 1875–1881. AAAI Press (2014)
17. Kulis, B., Grauman, K.: Kernelized locality-sensitive hashing for scalable image search. In: IEEE International Conference on Computer Vision ICCV (2009)
18. Leake, D., Smyth, B., Wilson, D., Yang, Q. (eds.): Maintaining Case-Based Reasoning Systems. Blackwell (2001), special issue of *Computational Intelligence*, 17(2), 2001.
19. Leetaru, K., Schrodt, P.A.: Gdelt: Global data on events, location, and tone. ISA Annual Convention (2013)
20. Lin, Y.B., Ping, X.O., Ho, T.W., Lai, F.: Processing and analysis of imbalanced liver cancer patient data by case-based reasoning. In: The 7th 2014 Biomedical Engineering International Conference. pp. 1–5 (Nov 2014)
21. Malof, J., Mazurowski, M., Tourassi, G.: The effect of class imbalance on case selection for case-based classifiers: An empirical study in the context of medical decision support. Neural Networks 25, 141–145 (2012)
22. Meng, X., Bradley, J.K., Yavuz, B., Sparks, E.R., Venkataraman, S., Liu, D., Freeman, J., Tsai, D.B., Amde, M., Owen, S., Xin, D., Xin, R., Franklin, M.J., Zadeh, R., Zaharia, M., Talwalkar, A.: Mllib: Machine learning in apache spark. CoRR abs/1505.06807 (2015)
23. Mühleisen, H., Bizer, C.: Web data commons - extracting structured data from two large web corpora. In: Bizer, C., Heath, T., Berners-Lee, T., Hausenblas, M. (eds.) WWW2012 Workshop on Linked Data on the Web, Lyon, France, 16 April, 2012. CEUR Workshop Proceedings, vol. 937. CEUR-WS.org (2012)
24. Ontañón, S., Plaza, E.: Collaborative case retention strategies for CBR agents. In: Case-Based Reasoning Research and Development: Proceedings of the Fifth International Conference on Case-Based Reasoning, ICCBR-03. Springer-Verlag, Berlin (2003)
25. Palmer, C.R., Faloutsos, C.: Density biased sampling: An improved method for data mining and clustering. SIGMOD Rec. 29(2), 82–92 (May 2000)
26. Rojas, J.A.R., Kery, M.B., Rosenthal, S., Dey, A.: Sampling techniques to improve big data exploration. In: 2017 IEEE 7th Symposium on Large Data Analysis and Visualization (LDAV). pp. 26–35 (Oct 2017)
27. Salamó, M., López-Sánchez, M.: Adaptive case-based reasoning using retention and forgetting strategies. Know.-Based Syst. 24(2), 230–247 (Mar 2011)
28. Smyth, B., Cunningham, P.: The utility problem analysed: A case-based reasoning perspective. In: Proceedings of the Third European Workshop on Case-Based Reasoning. pp. 392–399. Springer, Berlin (1996)
29. Smyth, B., Keane, M.: Remembering to forget: A competence-preserving case deletion policy for case-based reasoning systems. In: Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence. pp. 377–382. Morgan Kaufmann, San Mateo (1995)
30. Smyth, B., McKenna, E.: Footprint-based retrieval. In: Case-Based Reasoning Research and Development, ICCBR 1999. pp. 343–357. Springer, Berlin (1999)
31. Upadhyaya, S.R.: Parallel approaches to machine learninga comprehensive survey. Journal of Parallel and Distributed Computing 73(3), 284 – 292 (2013), models and Algorithms for High-Performance Distributed Data Mining
32. Zhu, J., Yang, Q.: Remembering to add: Competence-preserving case-addition policies for case base maintenance. In: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence. pp. 234–241. Morgan Kaufmann (1999)