Acquiring Customer Preferences from Return-Set Selections

L. Karl Branting

LiveWire Logic, Inc.
5500 McNeely Drive, Suite 102
Raleigh, NC 27612
branting@livewirelogic.com

Abstract. This paper describes LCW, a procedure for learning customer preferences by observing customers' selections from return sets. An empirical evaluation on simulated customer behavior indicated that an uninformed hypothesis about customer weights leads to low ranking accuracy unless customers place some importance on almost all features or the total number of features is quite small. In contrast, LCW's estimate of the mean preferences of a customer population improved as the number of customers increased, even for larger numbers of features of widely differing importance. This improvement in the estimate of mean customer preferences led to improved prediction of individual customer's rankings, irrespective of the extent of variation among customers and whether a single or multiple retrievals were permitted. The experimental results suggest that the return set that optimizes benefit may be smaller for customer populations with little variation than for customer populations with wide variation.

1 Introduction

A growing proportion of sales transactions are conducted over the Internet. In business-to-customer sales, an e-commerce site must perform the *inventory selection task*, which consists of eliciting each customer's requirements, finding the item (product or service) from the business's inventory that most nearly satisfies those requirements, presenting the item to the customer, and prompting the customer to either consummate the sale or refine the requirements. An item in the inventory that satisfies the customer's requirements at least as well as any other item in the inventory is *optimal* with respect to the inventory and requirements.

Case-based reasoning (CBR) is an increasingly popular paradigm for the inventory selection task [Wil99,WLW98,SRB99]. In contrast to standard database retrieval, which is restricted to exact matches, retrieval in CBR systems can involve partial matches ordered by the degree to which each product satisfies the customer's requirements. This permits an optimal item to be presented to the customer even when nothing in the inventory completely satisfies the customer's requirements.

While discrimination nets are sometimes used in CBR for case retrieval (e.g., [Kol84]), the most common technique in e-commerce applications of CBR is nearest-neighbor retrieval [WA95]. In this approach, inventory items are ordered by the similarity between customers' requirements and inventory items under a metric defined on a set of numeric or symbolic features that the requirements share with the inventory items.

The most common such metric is scaled Euclidean distance, e.g., for cases $c1$ and $c2$:

$$dist(c1, c2) = \sqrt{\sum_{f=1}^{n} w_f \times d(c1_f, c2_f)^2}$$

where $c1_f$ and $c2_f$ are the values of feature f for cases $c1$ and $c2$, w_f is the weight assigned to feature f, and $d(c1_f, c2_f)$ equals $c1_f - c2_f$ if feature f is numeric, 1 if f is symbolic and $c1_f = c2_f$, and 0 otherwise.

The customer is typically presented with a *return set* consisting of the rs most similar items. The customer can select any of the items in the return set or perform another retrieval with modified requirements.

Two evaluation criteria for the product selection task can be distinguished:

- **Satisfaction**, the degree to which the customer's requirements are satisfied by the best inventory item presented to the customer. Satisfaction is maximized when the return set contains an optimal item. Satisfaction is related to the recall criterion used in Information Retrieval (IR) research in that it is a function of the case quality, but (as in [BHK$^+$97]) differs in that it assigns no penalty for unretrieved cases.
- **Cognitive load** [SCTC90] imposed on the customer to find the best inventory item that was presented. Cognitive load is a function both of the number of actions performed by the customer and the number of alternatives from which each action is selected. This criterion is related to the precision criterion used in IR research, in that low precision increases the number of alternatives from which a selection must be made.

Benefit (discussed in greater detail below) is a function of both satisfaction and cognitive load, reflecting the relative importance of both components.

In principle, there is a trade-off between satisfaction and cognitive load: satisfaction can be maximized at the expense of cognitive load by presenting the customer with all inventory items; similarly, cognitive load can be maximized at the cost of satisfaction by presenting a single choice. In practice, however, e-commerce customers tolerate only a very low cognitive load. Faced with repeated retrievals or lengthy lists of items for perusal, customers quickly abandon the site and try somewhere else [Nie00]. Thus, cognitive load must be kept low, which requires minimizing (1) the number of retrievals that the customer must perform to find an optimal item, (2) the size of each return set, and (3) the rank of the best item within each return set. Of these, the first is the most important, since each retrieval entails a network delay and an increase by the return-set size in the number of alternatives that the customer must consider.

The number of retrievals can be minimized by making the probability that an optimal item is in the initial return set as high as possible. For a given return-set size, this probability depends on how accurately the weights used in the similarity metric model the customer's preferences, i.e., the relative importance the customer attaches to the case features. If the customer's preferences can be modeled by a scaled Euclidean distance metric and the system has exact values for the feature weights, then the inventory items can be ranked perfectly. Under these circumstances, even a single-item return set can be guaranteed to contain the optimal item. In practice, however, uncertainty about feature weights means that a larger return set—and therefore larger cognitive load—is required to make it probable that an optimal item is presented.

Various approaches have been used for acquiring individual preferences. One approach is to interview the user to determine pairwise relative feature importance [KR93,Bra99]. Less intrusive approaches, such as collaborative filtering, attempt to infer preferences from *a priori* knowledge, such as group membership [GNOT92]. A third approach strives to form user models based on observations of user decisions, either in response to system suggestions ("candidate/revision" or "learn-on-failure" [BB97,Mae94]) or through passive observations [DBM+92].

This work explores the feasibility of learning customer preferences by observing customers' selections from return sets. This approach is appropriate for customer populations that share preferences to some extent. In such customer populations, two sources of uncertainty concerning an individual customer's preferences can be distinguished: uncertainty about the mean preferences of the entire customer population; and uncertainty about the individual's deviation from the population's mean on each feature. When individual variations are limited, mean customer preferences constitute a good model of most customers. Under these circumstances, a small return set is likely to contain an optimal item. The greater the variation, the larger the return set required to ensure a high-satisfaction case, and the higher the resultant cognitive load.

Mean customer preferences can be determined from a representative collection of individual customer's preferences. Individual customers' feature weights can, in turn, be estimated by observing individual return-set selections in the following manner: whenever the customer selects an item other than the item highest ranked using the current feature weights, the system adjusts the weights to make the selected item rank first. If the customer makes repeated retrievals, the weights can be repeatedly adjusted. The mean customer preferences can be estimated from the mean of the observed individual preferences.

This paper proposes a procedure for adjusting feature weights based on return-set selections and demonstrates the performance of the algorithm through experiments performed on simulated data. The experiments are intended to address the following questions:

- What is the probability of misranking an inventory item as a function of feature-weight error? Stated differently, for a given feature weight error, how large must the return set be to guarantee an acceptable probability that it will contain an optimal item?

– For a given customer variance and return-set size, how many customers must be seen to achieve an acceptable probability that the return set will contain an optimal item?
– How much difference is there in the learning rate when each customer performs only a single retrieval as opposed to repeated retrievals?
– How can the best trade-off between satisfaction and cognitive load be achieved?

In the experiments described below, several simplifying assumptions are made concerning the nature of the inventory-selection task. The inventory is assumed to consist of I items, each represented by n features with real values uniformly distributed across the interval [0..1]. The mean feature weights of the entire customer population are represented by an n-dimensional feature vector GW (meaning *global weights*), with all features weights normalized to the [0..1] interval. The preferences of the i^{th} customer are represented by a normalized n-dimensional feature vector, c_i. Each customer's feature weights are intended to represent the relative importance of each feature to that customer. The weights of each customers' preferences are normally distributed around the mean with some standard deviation σ[1].

The next section describes an experiment designed to gauge the importance of feature-weight accuracy. Section 3 describes LCW, a procedure for learning global weights from individual return-set selections. Section 4 then presents an experimental evaluation designed to determine the return-set size and minimum number of customers needed to achieve an acceptable probability that the return set will contain an optimal item for different values of σ.

2 The Importance of Feature-Weight Accuracy

This section investigates the degree of benefit that can be expected when a system lacks any knowledge of customers' preferences, that is, when the system does not perform any learning. An important variable in determining the importance of customer feature-weight accuracy is the actual distribution of each customer's feature weights. Intuitively, one would expect a minority of features to be of high importance, and a larger number of features to be less important (e.g., in car purchasing `price` may be very important, whereas `seat-cover material` is less important, though not inconsequential).

This distribution of feature importances, and four alternatives, can be represented by the following n-dimensional weight vectors:

1. *Two-Level.* For each feature there is 25 per cent probability that the feature weight is in the interval [0.75..1.0] and a 75 per cent probability that it is in the interval [0.0..0.25].

[1] The distribution is truncated to fit within the [0..1] interval, e.g., for global weight 0.8, no values above 1.0 are permitted, to restrict values to the [0..1] interval, and no values below 0.6 are permitted, so that the samples are symmetrical around the global weight.

2. *Exponential.* For $0 <= i < n$, exactly one feature has weight $\sqrt{0.75 * \frac{1}{2^i}}$.
3. *Linear.* For $0 <= i <= n-1$, exactly one feature has weight $1 - \frac{1+i}{1+n}$.
4. *Random.* Feature weights are uniformly distributed on the [0..1] interval.
5. *Equal.* All features are weighted equally.

The Two-Level distribution is intended to mimic human preferences by making a minority of features quite important and the majority of features less important, but not completely irrelevant. However, the assumption that the Two-Level distribution is the most accurate model is mere surmise that must be tested in experiments involving human subjects. It seems possible that the distribution of feature importances is context-dependent, varies among individuals, or depends on other features not yet identified. Moreover, the division of 25% more important, 75% less important, is arbitrary.

The four alternative feature weight distributions are included because of the uncertainty about the nature of human preferences. The Exponential distribution corresponds to having one very important feature, several moderately important features, and the remaining features of little importance. Linear and Random, by contrast, correspond to situations in which there is no clear division between important and unimportant features. Finally, Equal distribution corresponds to the assumption that all weights are equally important.

In each round of this experiment, an inventory of 100 items was created, together with a weight vector of one of the five distribution types, representing the actual preferences of a simulated customer, and a weight vector of random values, the *hypothesized weights*, representing an uninformed model of the customer. Finally, an additional item was created, representing the features requested by customer. This item is termed the *requirements* vector, r.

The behavior of an e-commerce site was simulated by sorting the inventory I by similarity to the requirements vector r under the hypothesized weights, HW, producing a *hypothesized ranking.* Next, the inventory item that best matched the requirements under the actual weights (i.e., the optimal item) was selected[2]. Finally, the position of the optimal item in the hypothesized ranking was determined and the cosine between the actual and hypothesized weight vectors was calculated.

One thousand rounds were performed for 2, 4, 8, 16, and 32 features for each distribution. In each trial three values were calculated:

- The mean rank of the optimal item in the hypothesized ranking.
- The mean cosine between the actual and hypothesized weight vectors.
- The minimum return-set size needed to included at least 90 per cent of the optimal cases.

Figures 1 and 2 display the mean cosine and minimum size needed to included at least 90 per cent of the optimal cases, respectively. These figures show that as the number of features increases, the cosine between the actual customer weights

[2] With inventory items having random, real-valued weights, there is always a unique optimal item.

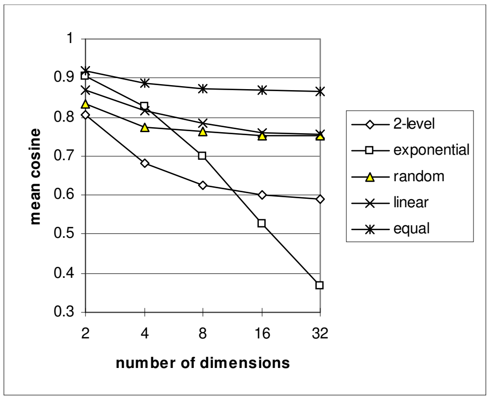

Fig. 1. The mean cosine between the actual and uninformed hypothesized weights.

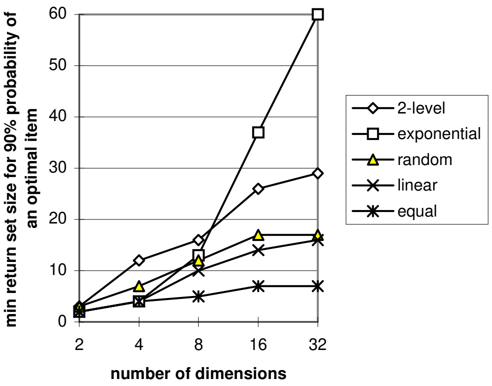

Fig. 2. The minimum return-set size needed to included at least 90 per cent of the optimal cases given uninformed weights.

and the hypothesized weights decreases and the minimum return-set size required for a 90 per cent probability of containing the optimal case increases. These changes, which correspond to decreased benefit, are more gradual for Random and Linear weight distributions than for Two-Level or Exponential. Intuitively, ranking accuracy decreases as the number of unimportant features increases[3].

These results indicate that unless customers' place some importance on almost all features or the total number of features is quite small, an uninformed hypothesis about customer weights can be expected to lead to low benefit. In other words, an unacceptably large return-set size is required to guarantee a high probability that the optimal case is returned. The next section presents an algorithm for modifying an initial hypothesis based on customer observations.

3 LCW: Learning Customer Weights

The LCW (Learning Customer Weights) algorithm rests on the assumption that customer preferences can be modeled by weights normally distributed around a population-wide mean, GW. The population-wide mean is modeled by hypothesized global weights, HW, initialized to random values. HW is used as the initial hypothesis for each individual customer's weights, hc_i. These weights are adjusted based on the customer's behavior to form revised weights hc'_i. The hypothesized global weights, HW, are then revised in light of hc'_i.

LCW consists of three connected algorithms: LearnGlobalWeights, LearnCustomerWeights, and AdjustWeights. The first algorithm, LearnGlobalWeights, takes arguments inventory I and return-set size rs and revises the global hypothesis, HW, in light of the apparent preferences of each customer. When a small number of customers have been seen, HW is simply the most recent hc'_i. When enough customers have been seen, HW is assigned the value of the mean of the last 10% of customers seen[4].

```
Procedure LearnGlobalWeights(I,rs)
HW := n random values;
FOR EACH customer i DO
   hci := HW;
   hci' := LearnCustomerWeights(hci,I,rs);
   IF i - 1 (i.e., the number of customers previously seen) < 20
   THEN HW := hci'
   ELSE HW := the mean of the last 10% of the previous hc's;
```

[3] Very similar results are obtained using equal, rather than random, hypothesized weights, except that the equal-weight hypothesis leads to the optimal item always being ranked first and a cosine of 1.0.

[4] This approach speeds learning, because when few customers have been seen, each hc'_i is likely to much closer to GW than any previous customer weights and, therefore, than the mean of the previous customer weights. When many customer have been seen, however, the mean of the customers may differ from G by less than the variance between individual customers.

The function LearnCustomerWeights takes hypothesized customer weights hc_i, inventory I, and return-set size rs, elicits the customer's requirements, r, ranks I by similarity to r under weight vector hc_i, and presents the closest rs for selection by the customer. When the customer selects an inventory item s in the return set other than the item that is most similar to r under weights hc_i, hc_i's weights are incrementally adjusted until s is ranked higher than any other member of the return set (or until the threshold for the maximum number of weight adjustments is exceeded). The cycle of retrieval and weight adjustment is repeated until the case ranked highest under the customer's actual weights, c_i, is ranked first under the hypothesized weights, hc_i, (corresponding to a satisfied customer making a selection) or the threshold for the maximum number of retrievals is exceeded (corresponding to a customer losing patience and terminating the sales episode). In the experiments described below, this threshold is set to either 1 or 10.

```
Function LearnCustomerWeights(hci,I,rs)
REPEAT
  HypothesizedRankings := I sorted by similarity to r under hci;
  ReturnSet := the first rs members of HypothesizedRankings;
  s := the member of ReturnSet most similar to r under ci,
         customer i's actual weight vector;
  IF s is not the first element of ReturnSet
  THEN hci := AdjustWeights(s,r,ReturnSet,hci,
                            MaximumAdjustmentLimit);
UNTIL s is the first element of ReturnSet OR
      Retrievals > MaximumRetrievalLimit.
return hci;
```

AdjustWeights iterates through each element, mri (for "misranked item"), of the return set that is incorrectly ranked higher than the customer's selection. For each such item, the weight of each feature of the hypothesized customer weights, hc_i, whose value in the requirements vector, r, is closer in value to mri than to s is decremented by the learning rate, lr. Conversely, the weight of each feature of hc_i whose value in r is closer to s than to mri is incremented.

```
Function AdjustWeights(s,r,ReturnSet,hci)
Adjustments := 0;
WHILE Adjustments < MaximumAdjustmentLimit
  SortedReturnSet := ReturnSet sorted by hci;
  MisRanked := elements of SortedReturnSet ranked above s;
  IF MisRanked = {}
  THEN return hci
  ELSE FOR EACH item mri in MisRanked
        FOR EACH feature f
          IF diff(f(mri),f(r)) < diff(f(r),f(s))
          THEN f(hci) *= (1 + LearningRate)
```

```
        ELSE IF diff(f(mri),f(r)) > diff(f(r),f(s))
             THEN f(hci) *= (1 - LearningRate);
return hci;
```

In the experiments below, LearningRate was 0.01 and MaximumAdjustmentLimit was 1000. Intuitively, AdjustWeights "pulls" the customer's selection s towards the customer's requirements r while "pushing" each member of MisRanked away from r [BCS97,ZY99]. Since AdjustWeights hill-climbs on a linear weight vector, it may not converge if the customer's actual preference function is not a linear function of item feature-values.

4 Experimental Evaluation

LCW induces an estimate of the mean preferences of a customer population based on estimations of each individual customer derived from that customer's behavior. Section 2 provided evidence that, absent knowledge about customers' preferences, ranking can be expected to be inaccurate if there are wide variations in feature importances unless the number of features is very low.

LCW would constitute an improvement over an uninformed weighting scheme if it satisfied the following two conditions: (1) LCW's estimate of the mean preferences of the customer population improved as the number of customers who have been seen increases, and (2) the improvement in the estimate of mean customer preferences resulted in an improvement in the accuracy of predictions of individual customers' rankings.

In the following experiment, the number of dimensions was held constant at 8, on the assumption that customers are unlikely to have the patience to enter more than 8 feature values. The inventory size was held constant at 100 items. In each round of the experiment, a new inventory was created, each item of which was an element of $[0..1]^8$, and a set of actual global weights was created, consisting of 8 values drawn from a two-level distribution. A set of 10, 20, 50, 100, or 250 customer preferences were created, each consisting of an 8-tuple of real numbers whose values were normally distributed around the global weights with $\sigma \in \{0, 0.1, 0.25, 0.5, 1\}$. Each customer was assigned a requirements vector r consisting of a random 8-tuple of real numbers. The maximum permitted retrievals was either 1 or 10, and the return-set size was $\in \{1, 2, 3, 5, 7, 10, 12, 15, 25, 50, 100\}$.

In each round, the global hypothesis, HW, was initially assigned random values, and LCW was applied to each simulated customer in turn. Fifty rounds were performed for each inventory, set of customers, σ, maximum retrievals, and return-set size.

4.1 Learning Global Weights

Figure 3 displays the mean cosine between the vectors representing the actual and hypothesized global weights as a function of the number of customers seen.

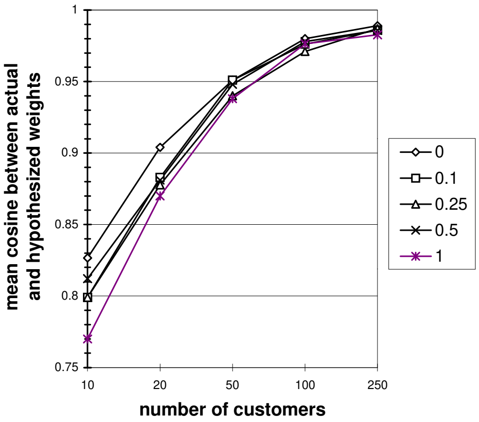

Fig. 3. The mean cosine between the actual and hypothesized global weights as a function of the number of customers seen, for σ from 0 to 1.0.

Separate learning curves are shown for σ from 0 to 1.0. The learning curve is steeper for customer populations with lower σ. However, the cosine is over .90 even for the largest σ tested after 100 customers have been seen. This indicates that the first condition mentioned above—that LCW's estimate of the mean preferences of the customer population improves as the number of customers who have been seen increases—is satisfied for a wide range of σs.

4.2 Improving Ranking

Evidence that the second condition—the improvement in the estimate of mean customer preferences leads to improved accuracy in predictions of individual customers' rankings—is shown in Figure 4, which shows that the minimum return-set size needed for a 90 per cent probability that the optimal case is in the return set decreases as the number of customers seen increases. For even the largest σ tested, 1.0, a return-set size of five is sufficient after 250 customers have been seen.

A different view of the experimental data is shown in Figure 5, which graphs the probability that the optimal item is in the return set as a function of return-set size, for customers seen of 10, 20, 50, 100, and 200 and σ of .25. The graph shows that when only 10 customers have been seen, there is a less than 90 per cent probability that the optimal case is in the return set for return-set size of less than 25. After 250 customers, however, a return-set size of 5 is sufficient.

4.3 Single vs. Multiple Retrievals

The experiment included two retrieval conditions. Under the first condition, each customer makes only a single retrieval. Alternatively, the customer makes

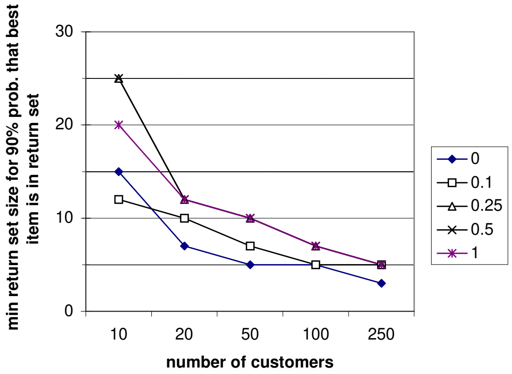

Fig. 4. The minimum return-set size needed for a 90 per cent probability that the optimal case is in the return set as a function of the number of customers, for σ from 0 to 1.0. The graph for σ of 0.5 overlaps the graph for σ of 0.25.

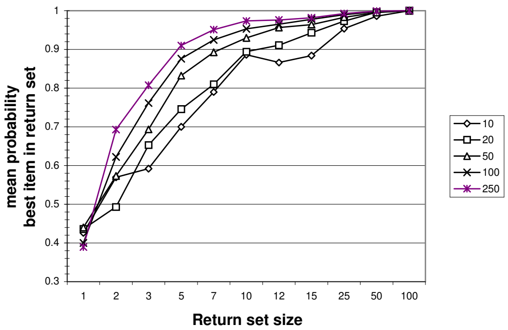

Fig. 5. The probability that the optimal item is in the return set as a function of return-set size, for 10, 20, 50, 100, and 200 customers and σ of 0.25.

repeated retrievals until the best item in the return set is ranked first up to 10 retrievals. The latter condition is intended to model relevance feedback, as in “more like this” retrievals that are permitted by some search engines. The observation above that customers seldom tolerate interactions requiring heavy cognitive load suggests that multiple retrievals, even if permitted, may be infrequent in practice.

How much opportunity for learning is lost when each customer is permitted only a single retrieval? Figure 6 compares the single versus multiple retrieval conditions by the mean probability that the optimal item is in the return set as a function of return-set size. The difference in performance under the two conditions is negligible. This reflects the fact (not shown in the graph) that the mean number of retrievals never exceeded 1.5 under any of the experimental conditions. This indicates that when a customer’s estimated weights have been adjusted once, further adjustments are not likely to produce an improvement as to that customer and inventory set.

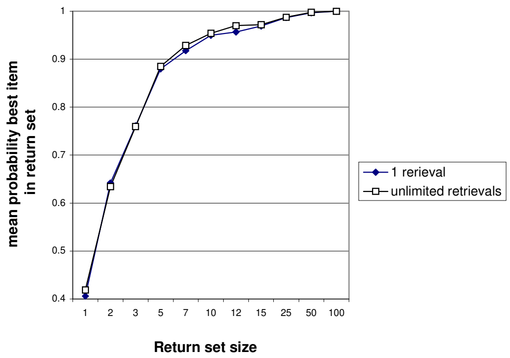

Fig. 6. The mean probability that the optimal item is in the return set as a function of return-set size, for 100 customers and σ of 0.25.

4.4 Maximizing Benefit

Choosing the optimal return-set size for a given customer population requires striking a balance between satisfaction and cognitive load. *Benefit* is intended to express the net trade-off between these two factors. The precise relative importance that should be attached to these two factors is impossible to determine *a priori*, as it depends on both the cognitive-load tolerance of the customer population and the importance that the vendor attaches to guaranteeing that an optimal case is returned. However, several heuristics for balancing satisfaction and cognitive load can be identified.

One heuristic is that an optimal case should almost always be returned. Under this heuristic, benefit is maximized by finding the smallest return set that insures a high probability of optimality. Another heuristic is that a return set of 1, which permits no learning, is about as bad as a return set consisting of all the cases, which maximizes cognitive load.

The tradeoff between satisfaction and cognitive load can be expressed in the following equation, in which k is a constant representing the relative importance of satisfaction as compared to cognitive load, rs is the return-set size, and sat represents satisfaction:

$$\text{benefit} = k \cdot sat - rs \cdot (1 - k)$$

Satisfaction, sat, can in turn be quantified by how much of an improvement the best item in the return set is over a randomly selected item. More precisely, sat can be measured by how much larger is (1) the mean distance between the requirements vector and every case in the inventory than (2) the distance between the requirements vector and the best item in the return set. Sat is normalized to the [0..1] interval by dividing it by the amount by which the optimal item is an improvement over a randomly selected item (more precisely, by the mean distance between the requirement vector and every case in the inventory minus the distance between the requirements vector and the optimal item):

$$sat = \frac{MeanDistance - dist(r, best-in-return-set)}{MeanDistance - dist(r, optimal-item)}$$

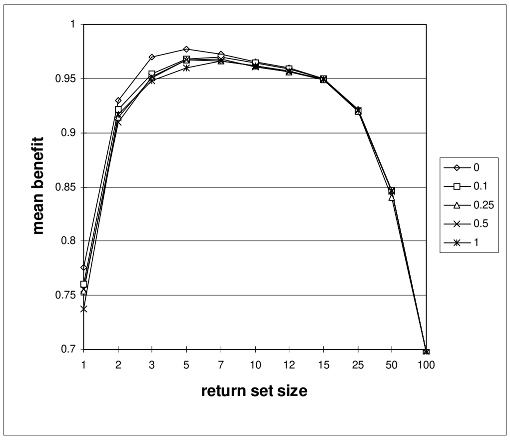

Fig. 7. Mean benefit as a function of return-set size for σ from 0 to 1.0.

If k is chosen to be .997, satisfaction is much more important than cognitive load, and the benefit is approximately the same for a return-set of size 1 as for a return set consisting of the entire inventory. Figure 7 displays the mean benefit as a function of return-set size for $k = .997$, 100 customers, and σ from 0 to 1.0. For small return-set sizes, the benefit is greater for customer populations with lower σ. For large return-set size, however, σ has little effect on benefit. This is because when the return set is sufficiently large, the optimal case is almost always returned regardless of σ. For $\sigma = 0$, benefit is maximized by a return-set size of 5. For larger σ, benefit is maximized by a return-set of 7.

5 Conclusion

This paper has identified the task of acquiring customer preferences from return-set selections, a learning problem that can arise when case-based reasoning is applied to the inventory-selection task. The paper described LCW, a procedure for performing this task, and proposed a criterion for retrieval performance, *benefit*, which embodies both the degree to which the best item in the return set satisfies the customer's requirements and the cognitive load imposed on the customer to select that item.

An empirical evaluation simulating the behavior of a customer population indicated that unless customers' place some importance on almost all features or the total number of features is quite small, an uninformed hypothesis about customer weights leads to low benefit. Simulation of LCW showed that, under the experimental conditions:

- LCW's estimate of the mean preferences of the customer population improves as the number of customers who have been seen increases.
- This improvement in the estimate of mean customer preferences leads to improved ranking performance for individual customers. Specifically, as the number of customers seen increases, the minimum return-set size needed to insure that an optimal item is returned decreases.
- Ranking performance when only one retrieval is permitted is almost identical to ranking performance when multiple retrievals are permitted.
- For small return-set size, benefit is higher for low σ customer populations than for high σ customer populations. The data suggest that the return set that optimizes benefit may be lower for low than for high σ customer populations.

These conclusions must be qualified by uncertainty concerning the assumptions underlying the experimental evaluation, i.e., whether:

- Customer preferences can be represented by a linear weight vector.
- For the customer population as a whole, the mean weights are distributed roughly into two levels.
- The weights of the individual customers for each feature are normally distributed around the mean value for that feature.

LCW illustrates how unobstrusive learning methods can improve the performance of web-based applications by customizing their performance to customers' behavior. Techniques for learning by observation are likely to become an increasingly important aspect of e-commerce.

References

[BB97] K. Branting and P. Broos. Automated acquisition of user preferences. *International Journal of Human-Computer Studies*, 46:55–77, 1997.

[BCS97] A. Bonzano, P. Cunningham, and B. Smyth. Using introspective learning to improve retrieval in CBR: A case study in air traffic control. In *Proceedings of the Second International Conference on Case-Based Reasoning*, pages 291–302, Providence, Rhode Island, July 25–27 1997. Springer.

[BHK+97] R. Burke, K. Hammond, V. Kulyukin, S. Lytinen, N. Tomuro, and S. Schoenberg. Question answering from frequently-asked question files: Experiences with the FAQ finder system. Technical Report TR-97-05, University of Chicago, Department of Computer Science, 1997.

[Bra99] K. Branting. Active exploration in instance-based preference modeling. In *Proceedings of the Third International Conference on Case-Based Reasoning (ICCBR-99), Lecture Notes in Artificial Intelligence 1650*, Monastery Seeon, Germany, 1999.

[DBM+92] L. Dent, J. Boticario, J. McDermott, T. Mitchell, and D. Zabowski. A personal learning apprentice. In *Proceedings of Tenth National Conference on Artificial Intelligence*, pages 96–103, San Jose, CA, July 12–16 1992. AAAI Press/MIT Press.

[GNOT92] D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[Kol84] J. Kolodner. *Retrieval and Organizational Strategies in Conceptual Memory: a Computer Model.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1984.

[KR93] R. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs.* Cambridge University Press, second edition, 1993.

[Mae94] P. Maes. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):31–40, 1994.

[Nie00] J. Nielson. *Designing Web Usability.* New Riders Publishing, Indianapolis, Indiana, USA, 2000.

[SCTC90] J. Sweller, P. Chandler, P. Tierney, and M. Cooper. Cognitive load as a factor in the structuring of technical material. *Journal of Experimental Psychology: General*, pages 176–192, 1990.

[SRB99] M. Schumacher and T. Roth-Berghofer. Architecture for integration of CBR systems with databases for e-commerce. In *Proceedings of the Seventh German Workshop on CBR (GWCBR'99)*, 1999.

[WA95] D. Wettschereck and D. Aha. Weighting features. In *Lecture Notes in Artificial Intelligence*, pages 347–358, Sesimbra, Portugal, October 1995. Springer.

[Wil99] W. Wilke. *Knowledge Management for Intelligent Sales Support in Electronic Commerce.* PhD thesis, University of Kaiserslautern, 1999.

[WLW98] W. Wilke, M. Lenz, and S. Wess. Intelligent sales support with CBR. In M. Lenz, B. Bartsch-Spoerl, H.-D. Burkhard, and S. Wess, editors, *Case-Based Reasoning Technology: from Foundations to Applications. LNAI 1400*, volume 1400, pages 91–113. Springer, 1998.

[ZY99] Z. Zhang and Q. Yang. Dynamic refinement of feture weights using quantitative introspective learning. In *International Joint Conference on Artificial Intelligence*, pages 228–233. Morgan Kaufmann, August 1999.