Some Formal Analysis of Rocchio's Similarity-Based Relevance Feedback Algorithm *

Zhixiang Chen (chen@cs.panam.edu)
Department of Computer Science, University of Texas-Pan American, 1201 West University Drive, Edinburg, TX 78539, USA.

Binhai Zhu (bhz@cs.montana.edu)
Department of Computer Science, Montana State University, Bozeman, MT 59717, USA.

June 16, 2001

Abstract. Rocchio's similarity-based Relevance feedback algorithm, one of the most important query reformation methods in information retrieval, is essentially an adaptive supervised learning algorithm from examples. In spite of its popularity in various applications there is little rigorous analysis of its learning complexity in literature. In this paper we show that in the binary vector space model, if the initial query vector is $\mathbf{0}$, then for any of the four typical similarities (inner product, dice coefficient, cosine coefficient, and Jaccard coefficient), Rocchio's similarity-based relevance feedback algorithm makes at least n mistakes when used to search for a collection of documents represented by a monotone disjunction of at most k relevant features (or terms) over the n-dimensional binary vector space $\{0,1\}^n$. When an arbitrary initial query vector in $\{0,1\}^n$ is used, it makes at least $(n+k-3)/2$ mistakes to search for the same collection of documents. The linear lower bounds are independent of the choices of the threshold and coefficients that the algorithm may use in updating its query vector and making its classification.

Keywords: relevance feedback, vector space, supervised learning, similarity, lower bound.

1. Introduction

Research on relevance feedback in information retrieval has a long history (Baeza-Yates and Ribeiro-Neto, 1999; Frakes and Baeza-Yates, 1992; Ide, 1971b; Ide, 1971a; Raghavan and Wong, 1986; J.J. Rocchio, 1971; Salton, 1989). It is regarded as the most popular query reformation strategy (Baeza-Yates and Ribeiro-Neto, 1999). The central idea of relevance feedback is to improve search performance for a particular query by modifying the query step by step, based on the user's judgments of the relevance or irrelevance of some of the documents retrieved. In the vector space model (Salton, 1989; Salton et al., 1975),

* The extended abstract of this paper was published in *Proceedings of the Eleventh International Symposium on Algorithms and Computation (ISAAC'00), Lecture Notes in Computer Science 1969*, pages 108-119, Springer-Verlag, December, 2000.

both documents and queries are represented as vectors in a discretized vector space. In this case, relevance feedback is essentially an adaptive supervised learning algorithm: A query vector and a similarity are used to classify documents as relevant and irrelevant; the user's judgments of the relevance or irrelevance of some the classified documents are used as examples for updating the query vector as a linear combination of the initial query vector and the examples judged by the user. Especially, when the inner product similarity is used, relevance feedback is just a Perceptron-like learning algorithm (Lewis, 1991). It is known (J.J. Rocchio, 1971) that there is an optimal way for updating the query vector if the sets of relevant and irrelevant documents are known. Practically it is impossible to derive the optimal query vector, because the full sets of the relevant and irrelevant documents are not available.

There are many different variants of relevance feedback in information retrieval. However, in this paper we only study Rocchio's similarity-based relevance feedback algorithm (J.J. Rocchio, 1971; Ide, 1971a; Salton, 1989). In spite of its popularity in various applications, there is little rigorous analysis of its complexity as a learning algorithm in literature. This is the main motivation for us to investigate the learning complexity of Rocchio's similarity-based relevance feedback algorithm. Wong, Yao and Bollmann (Wong et al., 1988) studied the linear structure in information retrieval. They designed a very nice gradient descent procedure to compute the coefficients of a linear function and analyzed its performance. In order to update the query vector adaptively, their gradient descent procedure must know the user preference which is in practice the unknown target to be learned by an information retrieval system. More discussions of their gradient descent procedure will be given in section 5.1.

The main contribution of our work in this paper is that linear lower bounds on classification mistakes are proved for the algorithm when any of the four typical similarities (inner product, dice coefficient, cosine coefficient, and Jaccard coefficient) listed in (Salton, 1989) is used. Technically, our work in this paper is enlightened by the work in (Kivinen et al., 1997) on lower bounds of the Perceptron algorithm. Precisely, we borrow the method developed in (Kivinen et al., 1997) for constructing an example sequence with pairwise constant inner products. We extend the method to cope with other similarities besides inner product. We also design a new method for selecting trial sequences and prove in a uniform way our lower bounds for Rocchio's similarity-based relevance feedback algorithm.

It should be pointed out that the lower bounds established in this paper for Rocchio's similarity-based relevance feedback algorithm is based on the following worst case considerations: The user acts as an

adversary to the algorithm; the algorithm is required to precisely search for the collection of all documents relevant to the given search query; and the algorithm is allowed to receive one document example judged by the user as relevance or irrelevant at each step [1]. In practical applications , in contrast to the above worst case considerations, the user in general may not act as an adversary to the algorithm; the algorithm is usually required to search for a short list of top ranked documents relevant to the given search query; and at each step of the similarity-based relevance algorithm, the user may judge a few documents as relevance feedback to the algorithm. In other words, the appropriate situation in real-world information retrieval applications would be a kind of *"sympathetic oracle"* model, where the user is not an adversary to the information retrieval system but a *"sympathetic judge"* who provides the most useful possible information in order to help the system help him/her to accomplish his/her work. Hence, our lower bounds proved in this paper for Rocchio's similarity-based relevance feedback algorithm *may not affect* the algorithm's effective applicability to the real-world problems despite of their theoretical significance. The formal analysis of the algorithm helps us understand the nature of the algorithm well so that we may find new strategies to improve its effectiveness or design new algorithms for information retrieval. Recently, we have made some progress along this line in (Chen, 2001). We have designed two types of multiplicative adaptive algorithms for user preference retrieval with provable better performance: One has better performance than Rocchio's algorithm in learning a class of linear classifiers over non-binary vector space. The other boosts the usefulness of an index term exponentially, while the gradient descent procedure in (Wong et al., 1988) boosts the usefulness of an index term linearly.

We refer the readers to the work of (Salton and Buckley, 1990; Lewis, 1991) for discussions of the Perceptron-like learning nature of the similarity-based relevance feedback algorithm. It was stated in (Lewis, 1991) that the most important future direction for research in information retrieval is likely to be machine learning techniques which can combine empirical learning with the use of knowledge bases.

This paper is organized as follows. In section 2, we give a formal presentation of Rocchio's similarity-based relevance feedback algorithm. In section 3, we prove several technical lemmas based on the techniques developed in (Kivinen et al., 1997). In section 4, we prove linear lower bounds for Rocchio's similarity-based relevance feedback algorithm with any of the four typical similarities listed in (Salton, 1989).

[1] This last restriction is not critical to the proof of the lower bounds, but it would make the analysis easier.

In section 5 we give some discussions of our result and also show that Rocchio's algorithm can be used to learn many other target document classes. We conclude the paper and list several open problems in section 6.

2. Rocchio's Similarity-Based Relevance Feedback Algorithm

Let R be the set of all real values, and let R^+ be the set of all non-negative real values. Let n be a positive integer. In the binary vector space model in information retrieval (Salton, 1989; Salton et al., 1975), a collection of n features or terms $T_1, T_2, \ldots, T_n$ are used to represent documents and queries. Each document $\mathbf{d}$ is represented as a vector $v_{\mathbf{d}} = (d_1, d_2, \ldots, d_n)$ such that for any i, $1 \leq i \leq n$, the i-th component of $v_{\mathbf{d}}$ is one if the i-th feature T_i appears in $\mathbf{d}$ or zero otherwise. Each query $\mathbf{q}$ is represented by a vector $v_{\mathbf{q}} = (q_1, q_2, \ldots, q_n)$ such that for any i, $1 \leq i \leq n$, the i-th component of $v_{\mathbf{q}} \in R$ is a real value used to determine the relevance (or weight) of the i-th feature T_i. Because of the unique vector representations of documents and queries, for convenience we simply use $\mathbf{d}$ and $\mathbf{q}$ to stand for their vector representations $v_{\mathbf{d}}$ and $v_{\mathbf{q}}$, respectively.

A similarity in general is a function m from $R^n \times R^n$ to R^+. A similarity m is used to determine the *relevance closeness* of documents to the search query and to rank documents according to such closeness. In the binary vector space model of information retrieval (Salton, 1989; Salton et al., 1975; Baeza-Yates and Ribeiro-Neto, 1999), to retrieve relevant documents for a given query vector $\mathbf{q}$ with respect to a similarity m, the system searches for all the documents $\mathbf{d}$, classifies those with similarity values $m(\mathbf{q}, \mathbf{d})$ higher than an explicit or implicit threshold as relevant, and returns to the user a short list of relevant documents with highest similarity values. This information retrieval process is in fact determined by a linear classifier, as defined later in this section, which is composed of a query vector $\mathbf{q}$, a similarity m, and a real-valued threshold ψ.

Unfortunately, in the real-world information retrieval applications, usually an ideal query vector cannot be generated due to many factors such as the limited knowledge of the users about the whole document collection. A typical example is the real-world problem of web search. In such a case, the user may use a few keywords to express what documents are wanted. However, it is nontrivial for both the user and a web search engine to *precisely* define the collection of documents wanted as a query vector composed of a set of keywords. The alternative

solution to the query formation problem is, as stated in (Salton, 1989), to conduct searches iteratively, first operating with a tentative query formation (i.e., an initial query vector), and then improving formations for subsequent searches based on evaluations of the previously retrieved materials. This type of methods for automatically generating improved query formation is called relevance feedback, and one particular and well-known example is Rocchio's similarity-based relevance feedback (J.J. Rocchio, 1971; Ide, 1971a; Salton, 1989).

Rocchio's similarity-based relevance feedback algorithm works in a step by step adaptive refinement fashion as follows. Starting at an initial query vector $\mathbf{q}_1$, the algorithm searches for all the documents $\mathbf{d}$ such that $\mathbf{d}$ is *very close* to $\mathbf{q}_1$ according to the similarity m, ranks them by $m(\mathbf{q}, \mathbf{d})$, and finally presents a short list of the top ranked documents to the user. The user examines the returned list of documents and judges some of the documents as relevant or irrelevant. At step $t \geq 1$, assume that the list of documents the user judged is $\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}$. Then, the algorithm updates its query vector as $\mathbf{q}_t = \alpha_{t_0}\mathbf{q}_1 + \sum_{j=1}^{t-1} \alpha_{t_j}\mathbf{x}_j$, where the coefficients $\alpha_{t_j} \in R$ for $j = 0, 1, \ldots, t-1$. At step $t+1$, the algorithm uses the updated query vector $\mathbf{q}_t$ and the similarity m to search for relevant documents, ranks the documents according to m, and presents the top ranked documents to the user. In practice, a threshold θ is explicitly (or implicitly) used to select the highly ranked documents. Practically, the coefficients α_{t_j} may be fixed as $1, -1$ or 0.5 (Baeza-Yates and Ribeiro-Neto, 1999; Salton, 1989). The following four typical similarities were listed in (Salton, 1989): For any $\mathbf{q}, \mathbf{x} \in R^n$,

$$
\begin{aligned}
inner\ product: \ m_1(\mathbf{q}, \mathbf{x}) &= \sum_{i=1}^{n} q_i x_i, \\
dice\ coefficient: \ m_2(\mathbf{q}, \mathbf{x}) &= \frac{2m_1(\mathbf{q}, \mathbf{x})}{m_1(\mathbf{q}, \mathbf{q}) + m_1(\mathbf{x}, \mathbf{x})}, \\
cosine\ coefficient: \ m_3(\mathbf{q}, \mathbf{x}) &= \frac{m_1(\mathbf{q}, \mathbf{x})}{\sqrt{m_1(\mathbf{q}, \mathbf{q})}\sqrt{m_1(\mathbf{x}, \mathbf{x})}}, \\
Jaccard\ coefficient: \ m_4(\mathbf{q}, \mathbf{x}) &= \frac{m_1(\mathbf{q}, \mathbf{x})}{m_1(\mathbf{q}, \mathbf{q}) + m_1(\mathbf{x}, \mathbf{x}) - m_1(\mathbf{q}, \mathbf{x})}.
\end{aligned}
$$

To make the above definitions valid for arbitrary $\mathbf{q}$ and $\mathbf{x}$, we define that the similarity between two zero vectors is zero, i.e.,

$$m_i(\mathbf{0}, \mathbf{0}) \ = \ 0, \ for\ 1 \leq i \leq 4.$$

It should be pointed out that the cosine similarity m_3 is nothing but the inner product similarity m_1 when the vectors are normalized. It is also easy to show that Jaccard similarity m_4 is a strictly monotonic

transformation of the dice similarity m_2. This implies that both the inner product similarity and the cosine similarity may achieve equivalent classification for any document document with respect to the given query vector. The subtle, but important, difference between the two similarities is that different rank values may be obtained for a document with respect to the given query vector so that different "ranking gaps" are obtained between documents. For example, when the rank gap between documents $\mathbf{x}$ and $\mathbf{y}$ based on the inner product similarity m_1 is $m_1(\mathbf{q}, \mathbf{x}) - m_1(\mathbf{q}, \mathbf{y}) = 0.8$, the rank gap based on the similarity m_3 may be $m_3(\mathbf{q}, \mathbf{x}) - m_3(\mathbf{q}, \mathbf{y}) = 0.2$. Analogously, the above analysis applies to the dice similarity and the Jaccard similarity. Different rank gaps between pairs of documents based on different similarities define different user preference structures of the documents, hence different document groups or clusters may be obtained. Therefore, different information retrieval performances *might be* achieved for different similarities. In other words, if the performance of an information retrieval system is not concerned, then cosine and inner product similarities may be regarded as two equivalent similarities, and so may Jaccard and dice similarities. In (Wong et al., 1988) user preference was studied in terms of structures of *weak order* and, in particular, linear order. In designing an adaptive information retrieval system, the system performance is definitely not a negligible factor. To our best knowledge, we do not know any provable theoretical results about influences of the different similarities on the performance of the similarity-based relevance feedback. Our linear lower bounds proved in this paper tell us that in the worst case, the four different similarities have the same affect on the performance of Rocchio's relevance feedback. But we do not know what the result will be in the average case.

As stated in (Baeza-Yates and Ribeiro-Neto, 1999), the main advantage of the relevance feedback is its simplicity and good results. The simplicity is due to the fact that the modified term weights (query vector components) are computed directly from the set of retrieved documents. The good results are observed experimentally and are due to the fact that the modified query vector does reflect a portion of the intended query semantics.

The similarity-based relevance feedback algorithm is essentially an adaptive supervised learning algorithm from examples (Salton and Buckley, 1990; Lewis, 1991). The goal of the algorithm is to learn some unknown classifier to classify documents as relevant or irrelevant. The learning is performed by modifying (or updating) the query vector that serves as the hypothetical representation of the collection of all relevant documents. The method for updating the query vector is similar to the

Perceptron algorithm. We given the necessary formal definitions in the following.

DEFINITION 1. *Let m from $R^n \times R^n$ to R^+ be a similarity. A classifier with respect to m over the n-dimensional binary vector space $\{0,1\}^n$ is a triple $(\mathbf{q}, \psi, m)$, where $\mathbf{q} \in R^n$ is a query vector, and $\psi \in R$ is a threshold. The classifier $(\mathbf{q}, \psi, m)$ classifies any documents $\mathbf{d} \in \{0,1\}^n$ as relevant if $m(\mathbf{q}, \mathbf{d}) \geq \psi$ or irrelevant otherwise. The classifier $(\mathbf{q}, \psi, m)$ is called a* linear classifier *with respect to the similarity m, if m is a linear function from $R^n \times R^n$ to R^+.*

For simplicity, we may just call $(\mathbf{q}, \psi, m)$ *a classifier*, or *a linear classifier* when m is linear. The following are examples of classifiers:

$$\begin{aligned}
(\mathbf{q}_1, m_1, 10.6), &\quad \mathbf{q}_1 = (0, \ldots, 0);\\
(\mathbf{q}_2, m_2, 0.987), &\quad \mathbf{q}_2 = (1, \ldots, 1);\\
(\mathbf{q}_3, m_3, \frac{3}{n}), &\quad \mathbf{q}_3 = (1, 2, 3, \ldots, 0);\\
(\mathbf{q}_4, m_4, \frac{1}{\sqrt{n}}), &\quad \mathbf{q}_4 = (1.2, 2.3, \ldots, 0.6).
\end{aligned}$$

In particular, $(\mathbf{q}_1, m_1, 10.6)$ is a linear classifier but the other three are not, because m_1 is linear and m_i are not linear for $i = 2, 3$ and 4.

DEFINITION 2. *An adaptive supervised learning algorithm A for learning a target classifier $(\mathbf{q}, \psi, m)$ over the n-dimensional binary vector space $\{0,1\}^n$ from examples is a game played between the algorithm A and the user in a step by step fashion, where the query vector $\mathbf{q}$ and the threshold ψ are unknown to the algorithm A, but the similarity m is. At any step $t \geq 1$, A gives a classifier $(\mathbf{q}_t, \psi_t, m)$ as a hypothesis to the target classifier to the user, where $\mathbf{q}_t \in R^n$ and $\psi_t \in R$. If the hypothesis is equivalent to the target, then the user says "yes" to conclude the learning process. Otherwise, the user presents an example $\mathbf{x_t} \in \{0,1\}^n$ such that the target classifier and the hypothesis classifier differ at $\mathbf{x_t}$. In this case, we say that the algorithm A makes a mistake. At step $t+1$, the algorithm A constructs a new hypothetical classifier $(\mathbf{q}_{t+1}, \psi_{t+1}, m)$ to the user based on the received examples $\mathbf{x_1}, \ldots, \mathbf{x_t}$. The learning complexity (or the mistake bound) of the algorithm A is in the worst case the maximum number of examples that it may receive from the user in order to learn some classifier.*

If the readers are familiar with on-line learning from equivalence queries (Angluin, 1987; Littlestone, 1988), then an adaptive supervised

learning algorithm as defined above is a proper on-line learning algorithm for learning the class of classifiers from equivalence queries over the n-dimensional binary vector space. We now give the formal definition of Rocchio's similarity-based relevance feedback algorithm.

DEFINITION 3. *Rocchio's similarity-based relevance feedback algorithm is an adaptive supervised learning algorithm for learning any classifier* $(\mathbf{q}, \psi, m)$ *over the* n*-dimensional binary vector space* $\{0,1\}^n$ *from examples. Let* $\mathbf{q}_1$ *be the initial query vector. At any step* $t \geq 1$*, the algorithm presents a classifier* $(\mathbf{q}_t, \psi_t, m)$ *as its hypothesis to the target classifier to the user, where* $\psi_t \in R$ *is the threshold, and the query vector* $\mathbf{q}_t$ *is modified as follows. Assume that at the beginning of step* t *the algorithm has received a sequence of examples* $\mathbf{x}_1, \ldots, \mathbf{x}_{t-1}$*, then the algorithm uses the following modified query vector* $\mathbf{q}_t$ *for its next classification:*

$$\mathbf{q}_t \;=\; \alpha_{t_0}\mathbf{q}_1 + \sum_{j=1}^{t-1} \alpha_{t_j}\mathbf{x}_j, \tag{1}$$

where $\alpha_{t_j} \in R$*, for* $j = 0, \ldots, t-1$*, are called additive updating factors.*

Please note that our definition above is a generalized version of Rocchio's original algorithm. In our definition, any function m from $R^n \times R^n$ to R^+ can be used as a similarity; arbitrary real values can be used in computing the updated query vector; and finally our definition allows adaptive learning until the target is obtained.

REMARK 4. *We would like to give the following remarks about Rocchio's similarity-based relevance feedback algorithm.*

(a) In the above definition, Rocchio's similarity-based relevance feedback algorithm can use any real-valued threshold and any real-valued additive updating factors at each step. But in practice the additive updating factors α_{t_j} *may be fixed as* 1 *(or* 0.5*) to promote the relevance of relevant examples and* -1 *(or* -0.5*) to demote the irrelevance of irrelevant examples (*Baeza-Yates and Ribeiro-Neto, 1999*;* Salton, 1989*).* α_{t_0} *is usually set to* 1*. Also in practice the threshold is usually implicitly used for selecting a short list of top ranked documents.*

(b) For the purpose of the worst case analysis of the mistake bounds of the algorithm, in the definition we only allow the algorithm to receive one example at each step. We consider that the user acts as an adversary to the algorithm and that the algorithm is required

to precisely learn (or search for) the target linear classifier. In practical applications, in contrast to the above worst case considerations, at each step the algorithm may receive several examples from the user; the user in general may not act as an adversary to the algorithm; and the algorithm is usually required to search for a short list of top ranked relevant documents. Hence, our lower bounds proved in this paper may not affect the algorithm's effective applicability to the real-world problems. The formal analysis of the algorithm helps us to understand its nature well so that we may be able to find new strategy to improve its effectiveness or to design new algorithms for information retrieval.

*(c) When the similarity m is the inner product of two vectors, then Rocchio's algorithm is similar to Rosenblatt's Perceptron algorithm (*Rosenblatt, 1958*).*

We will use the sets of documents represented by monotone disjunctions of relevant features to study the mistake bounds of Rocchio's algorithm. The efficient learnability of monotone disjunctions of relevant features (or attributes) has been extensively studied in machine learning (for example, (Littlestone, 1988)). Although very simple in format, monotone disjunctions are very common ways of expressing search queries, especially in the case of web search. All existing popular search engines support disjunctions of keywords as search query formations. For any k with $1 \leq k \leq n$, classifiers can be defined to precisely classify a monotone disjunction of at most k relevant features

$$x_{i_1} \vee \cdots \vee x_{i_s}, \ 1 \leq s \leq k. \tag{2}$$

i.e., to precisely classify whether any given document satisfies the monotone disjunction of (2) or not. If we choose a vector $\mathbf{u} \in R^n$ such that all its components are zero except that those at positions $i_1, \ldots, i_s$ are one, then it is easy to verify that for any $\mathbf{d} \in \{0,1\}^n$, each of the following four expressions is a necessary and sufficient condition for deciding whether $\mathbf{d}$ satisfies (2):

$$\begin{aligned} m_1(\mathbf{u},\mathbf{d}) &\geq \frac{1}{2}, \\ m_2(\mathbf{u},\mathbf{d}) &\geq \frac{2}{k+n}, \\ m_3(\mathbf{u},\mathbf{d}) &\geq \frac{1}{\sqrt{kn}}, \\ m_4(\mathbf{u},\mathbf{d}) &\geq \frac{1}{k+n-1}. \end{aligned}$$

This implies that $(\mathbf{u}, \frac{1}{2}, m_1)$, $(\mathbf{u}, 2/(k+n), m_2)$, $(\mathbf{u}, 1/\sqrt{kn}, m_3)$ and $(\mathbf{u}, 1/(k+n-1), m_4)$ are all respectively classifiers for (2).

3. Technical Lemmas

The technique used in (Kivinen et al., 1997) to prove linear lower bounds for the Perceptron algorithm (or, in general, linear additive on-line learning algorithms) is the construction of an example sequence $B = ((\mathbf{z'}_1, \mathbf{z''}_1), \ldots, (\mathbf{z'}_l, \mathbf{z''}_l))$ with pairwise constant inner products such that for any given initial query vector (or weight vector as used in (Kivinen et al., 1997)) and any linear classifier with the inner product similarity, if the initial query vector and the linear classifier differ on the sequence B, then the Perceptron algorithm makes one mistake at one of the two examples in every pair of B. In other words, each pair of the sequence B *preserves* the classification difference of the linear classifier and the initial query vector for the Perceptron algorithm when the examples in B are used to update the query vector. It was shown in (Kivinen et al., 1997) that row vectors of Hadamard matrices can be used to construct the required example sequence B. We will borrow the above technique from (Kivinen et al., 1997) to prove linear lower bounds for Rocchio's similarity-based relevance feedback algorithm. However, one must note that when a similarity (for example, the Jaccard coefficient similarity) other than the inner product similarity is used, the pairs of the sequence B as used in (Kivinen et al., 1997) *may not still preserve* the classification difference of the target classifier (which may not necessarily be linear) and the initial query vector. The rotation invariant concept (Kivinen et al., 1997) is in general not applicable to learning algorithms with a non-zero initial query vector, nor applicable to non-rotation variants of the linear additive learning algorithms. Therefore, we need to design new methods for constructing example sequences that are applicable to Rocchio's similarity-based relevance feedback algorithm with arbitrary initial query vector and any of the four similarities defined in section 2.

In the following we extend Definition 7 given in (Kivinen et al., 1997) to deal with any similarity.

DEFINITION 5. *Let the sequence* $B = ((\mathbf{z'}_1, \mathbf{z''}_1), \ldots, (\mathbf{z'}_l, \mathbf{z''}_l))$, *where* $\mathbf{z'}_t$ *and* $\mathbf{z''}_t$ *are in* $\{0,1\}^n$ *for all* t. *Let* $\mathbf{q}_1 \in R^n$ *be a query vector,* m *a similarity, and* $(\mathbf{u}, \psi, m)$ *a classifier. Define* $\mathbf{q}_t = \alpha_{t_0}\mathbf{q}_1 + \sum_{j=1}^{t-1} \alpha_{t_j}\mathbf{x}_j$, *for* $t = 1, \ldots, l$, *where* $\mathbf{x}_j \in \{\mathbf{z'}_j, \mathbf{z''}_j\}$ *and* $\alpha_{t_j} \in R$. *We say that the query vector* $\mathbf{q}_t$ *and the classifier* $(\mathbf{u}, \psi, m)$ *differ on the sequence* B

with respect to m if either

$$m(\mathbf{q}_t, \mathbf{z'}_t) \leq m(\mathbf{q}_t, \mathbf{z''}_t) \text{ and } m(\mathbf{u}, \mathbf{z'}_t) > \psi > m(\mathbf{u}, \mathbf{z''}_t),$$
$$m(\mathbf{q}_t, \mathbf{z'}_t) \geq m(\mathbf{q}_t, \mathbf{z''}_t) \text{ and } m(\mathbf{u}, \mathbf{z'}_t) < \psi < m(\mathbf{u}, \mathbf{z''}_t).$$

We now prove a weaker version of Lemma 8 in (Kivinen et al., 1997) in which only the initial query vector (or weight vector in their term) is required to differ from the target linear classifier on the example sequence, whereas we require that all the query vectors (the initial one and the updated ones) differ from the target classifier which in general may not be linear. We do not use the pairwise constant inner product property, because we may not have such a property for other similarities.

LEMMA 6. *Let m from $R^n \times R^n$ to R^+ be a similarity and let $\mathbf{q}_1 \in R^n$ be the initial query vector. Let the sequence*

$$B = ((\mathbf{z'}_1, \mathbf{z''}_1), \ldots, (\mathbf{z'}_l, \mathbf{z''}_l)),$$

where $\mathbf{z'}_t$ and $\mathbf{z''}_t$ are in $\{0,1\}^n$ for all t. For any classifier $(\mathbf{u}, \psi, m)$ over the domain $\{0,1\}^n$, if $\mathbf{q}_t$, which is as defined in Definition 3, and $(\mathbf{u}, \psi, m)$ differ on B with respect to m for $t = 1, \ldots, l$, then Rocchio's similarity-based relevance feedback algorithm makes at least l mistakes for learning the classifier $(\mathbf{u}, \psi, m)$.

Proof. Let A be Rocchio's similarity-based relevance feedback algorithm for learning $(\mathbf{u}, \psi, m)$. Consider the trial sequence

$$S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l)),$$

in which $\mathbf{x}_t \in \{z'_t, z''_t\}$, and y_t is the classification value of $\mathbf{x}_t$ determined by the unknown target classifier $(\mathbf{u}, \psi, m)$ for $t = 1, \ldots, l$. In other words, $y_t = 1$ if $(\mathbf{u}, \psi, m)$ classifies $\mathbf{x}_t$ as relevant or $y_t = 0$ otherwise. At any step t with $1 \leq t \leq l$, the hypothesis of the algorithm A is $(\mathbf{q}_t, \psi_t, m)$ with

$$\mathbf{q}_t = \alpha_{t_0}\mathbf{q}_1 + \sum_{j=1}^{t-1} \alpha_{t_j}\mathbf{x}_j$$

according to expression (1) of Definition 3. By the assumption, $\mathbf{q}_t$ and $(\mathbf{u}, \psi, m)$ differ on the sequence B with respect to m. That is, we have either $m(\mathbf{q}_t, \mathbf{z'}_t) \leq m(\mathbf{q}_t, \mathbf{z''}_t)$ and $m(\mathbf{u}, \mathbf{z'}_t) > \psi > m(\mathbf{u}, \mathbf{z''}_t)$, or $m(\mathbf{q}_t, \mathbf{z'}_t) \geq m(\mathbf{q}_t, \mathbf{z''}_t)$ and $m(\mathbf{u}, \mathbf{z'}_t) < \psi < m(\mathbf{u}, \mathbf{z''}_t)$. In the first case, if $\psi_t \leq m(\mathbf{q}_t, \mathbf{z}'_t)$, the adversary chooses $\mathbf{x}_t = \mathbf{z}''_t$. In this case, $y_t = 0$, i.e., the target classifier classifies $\mathbf{x}_t = \mathbf{z}''_t$ as irrelevant, but the hypothesis issued by A classifies it as relevant, thus A makes a

mistake. If $\psi_t > m(\mathbf{q}_t, \mathbf{z}'_t)$, the adversary chooses $\mathbf{x}_t = \mathbf{z}'_t$. In such a case, $y_t = 1$, i.e., the target classifier classifies $\mathbf{x}_t = \mathbf{z}'_t$ as relevant, but the hypothesis issued by A classifies it as irrelevant, thus again A makes a mistake. In the second case of $m(\mathbf{q}_t, \mathbf{z'}_t) \geq m(\mathbf{q}_t, \mathbf{z''}_t)$ and $m(\mathbf{u}, \mathbf{z'}_t) < \psi < m(\mathbf{u}, \mathbf{z''}_t)$, with the same manner we can show that the learning algorithm A makes a mistake at either $\mathbf{z}'_t$ or $\mathbf{z}''_t$. Therefore, A makes l mistakes on the trial sequence S. This means that A makes at least l mistakes for learning the unknown target classifier $(\mathbf{u}, \psi, m)$. □

We now follow the approach in (Kivinen et al., 1997) to construct example sequences from row vectors of Hadamard matrices. Such sequences are essentially applicable to Rocchio's similarity-based relevance feedback algorithm when the zero initial vector is used, but not when a non-zero initial vector is used. Let $\mathbf{I}_n$ be the identity matrix of order n.

DEFINITION 7. *A Hadamard matrix* $\mathbf{H}_n$ *of order* n *is an* $n \times n$ *matrix with elements in* $\{-1, 1\}$*, such that*

$$\mathbf{H}_n^T \mathbf{H}_n = \mathbf{H}_n \mathbf{H}_n^T = n\mathbf{I}_n. \tag{3}$$

$\mathbf{H}_n$ *is normalized if the first row and the first column consist of ones only.*

The above (2) implies that any two distinct rows (or columns) of a $\mathbf{H}_n$ are orthogonal. Normalized Hadamard matrices can be constructed as follows. Let $H_1 = (1)$. For any $n = 2^d$ with $d \geq 0$, define

$$H_{2n} = \begin{pmatrix} H_n & H_n \\ H_n & -H_n \end{pmatrix}. \tag{4}$$

Two examples of Hadamard matrices are given as follows.

$$H_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{pmatrix}.$$

$$H_8 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{pmatrix}.$$

The following property follows from (3) and (4).

PROPOSITION 8. *For $n = 2^d$ with $d > 0$, let h_t be the t-th row of the normalized Hadamard matrix H_n for $t = 1, \ldots, n$, we have*

$$m_1(h_i, h_t) = 0, \ for\ 1 \le i < t \le n, \tag{5}$$
$$m_1(h_i, h_i) = n, \ for\ 1 \le i \le n, \tag{6}$$
$$\sum_{j=1}^{n} h_{1j} = n, \ and\ \sum_{j=1}^{n} h_{ij} = 0\ for\ 1 < i \le n. \tag{7}$$

DEFINITION 9. *Let $n = 2^d + k - 1$ for some positive integers d and k. For $t = 1, \ldots, 2^d$, let $\mathbf{h}_t$ be the t-th row of the normalized Hadamard matrix H_{2^d}. We define B_H to be the sequence $((\mathbf{z}'_1, \mathbf{z}''_1), \ldots, (\mathbf{z}'_{2^d}, \mathbf{z}''_{2^d}))$, where*

$$\mathbf{z}'_t = ((h_{t,1} + 1)/2, \ldots, (h_{t,2^d} + 1)/2, 0, \ldots, 0),$$
$$\mathbf{z}''_t = ((-h_{t,1} + 1)/2, \ldots, (-h_{t,2^d} + 1)/2, 0, \ldots, 0)$$

PROPOSITION 10. *Let $n = 2^d + k - 1$ for some positive integers d and k and let $B_H = ((\mathbf{z}'_1, \mathbf{z}''_1), \ldots, (\mathbf{z}'_{2^d}, \mathbf{z}''_{2^d}))$ be the sequence as defined in Definition 9. For any i and j with $1 \le i, j \le 2^d$, we have*

(a) $m_1(\mathbf{z}'_1, \mathbf{z}'_1) = 2^d$; $m_1(\mathbf{z}''_1, \mathbf{z}''_1) = 0$;
$m_1(\mathbf{z}'_i, \mathbf{z}'_i) = m_1(\mathbf{z}''_i, \mathbf{z}''_i) = 2^{d-1}$, $if\ 1 < i \le 2^d$.

(b) $m_1(\mathbf{z}'_1, \mathbf{z}'_j) = m_1(\mathbf{z}'_1, \mathbf{z}''_j) = 2^{d-1}$;
$m_1(\mathbf{z}''_1, \mathbf{z}'_j) = m_1(\mathbf{z}''_1, \mathbf{z}''_j) = 0$, $if\ 1 < j \le 2^d$.

(c) $m_1(\mathbf{z}'_i, \mathbf{z}'_j) = m_1(\mathbf{z}'_i, \mathbf{z}''_j) = m_1(\mathbf{z}''_i, \mathbf{z}'_j)$
$= m_1(\mathbf{z}''_i, \mathbf{z}''_j) = 2^{d-2}$, $if\ 1 < i < j \le 2^d$.

Proof. We only prove (a), because (b) and (c) can be coped with similarly. By Proposition 8,

$$m_1(\mathbf{z}'_i, \mathbf{z}'_i) = \frac{1}{4}\sum_{s=1}^{2^d}(h_{is} + 1)^2 = \frac{1}{4}(\sum_{s=1}^{2^d} h_{is}^2 + 2\sum_{s=1}^{2^d} h_{is} + \sum_{s=1}^{2^d} 1)$$
$$= \frac{1}{4}(2^d + 2\sum_{s=1}^{2^d} h_{is} + 2^d) = \frac{1}{4}(2^{d+1} + 2\sum_{s=1}^{2^d} h_{is}).$$

If $i = 1$, then $\sum_{s=1}^{2^d} h_{is} = 2^d$, hence $m(\mathbf{z}'_1, \mathbf{z}'_1) = 2^d$. If $i \neq 1$, then $\sum_{s=1}^{2^d} h_{is} = 0$, hence $m_1(\mathbf{z}'_i, \mathbf{z}'_i) = 2^{d-1}$. Similarly,

$$m_1(\mathbf{z}''_i, \mathbf{z}''_i) = \frac{1}{4}\sum_{s=1}^{2^d}(-h_{is} + 1)^2 = \frac{1}{4}(\sum_{s=1}^{2^d} h_{is}^2 - 2\sum_{s=1}^{2^d} h_{is} + \sum_{s=1}^{2^d} 1)$$

$$= \frac{1}{4}(2^d - 2\sum_{s=1}^{2^d} h_{is} + 2^d) = \frac{1}{4}(2^{d+1} - 2\sum_{s=1}^{2^d} h_{is}).$$

If $i = 1$, then $\sum_{s=1}^{2^d} h_{is} = 2^d$, hence $m_1(\mathbf{z''}_1, \mathbf{z''}_1) = 0$. If $i \neq 1$, then $\sum_{s=1}^{2^d} h_{is} = 0$, hence $m_1(\mathbf{z''}_i, \mathbf{z''}_i) = 2^{d-1}$. □

We now introduce a new method for constructing example sequences that are applicable to Rocchio's similarity-based relevance feedback algorithm with any of the four similarities and an arbitrary initial query vector. We expand a Hadamard matrix by adding rows and columns with zeroes, and exchange rows and columns of the expanded matrix according to the initial query vector. The new method is given in the proof of Proposition 11.

PROPOSITION 11. *Given any $n = 2^d + k - 1$ with positive integers d and k, for any query vector $\mathbf{q} \in \{0,1\}^n$, there is a sequence $D(\mathbf{q}) = ((\mathbf{v'}_1, \mathbf{v''}_1), \ldots, (\mathbf{v'}_{2^{d-1}}, \mathbf{v''}_{2^{d-1}}))$ such that $\mathbf{v'}_t$ and $\mathbf{v''}_t$ are in $\{0,1\}^n$ for $t = 1, \ldots, 2^{d-1}$, and the sequence satisfies all the three properties given in Proposition 10 with each occurrence of 2^d replaced by 2^{d-1} and each occurrence of $\mathbf{z}$ replaced by $\mathbf{v}$, respectively. Furthermore, we have $m_1(\mathbf{q}, \mathbf{v'}_1) = 0$ if $\mathbf{q}$ has at least 2^{d-1} zero components or $m_1(\mathbf{q}, \mathbf{v'}_1) = 2^{d-1}$ otherwise; and*

$$m_1(\mathbf{q}, \mathbf{v'}_i) = 0, \ for\ i \neq 1;$$
$$m_1(\mathbf{q}, \mathbf{v''}_i) = 0, \ for\ all\ i.$$

Proof. Given any vector $\mathbf{q} = (q_1, \ldots, q_n) \in \{0,1\}^n$ with $n = 2^d + k - 1$ for positive integers d and k, we have

$$2^{d-1} = \frac{n-k+1}{2} \leq \frac{n}{2}.$$

This means that we can choose 2^{d-1} components of $\mathbf{q}$, denoted by $q_{i_1}, \ldots, q_{i_{2^{d-1}}}$, such that they are either all one or all zero. Define the $n \times n$ matrix C_n as follows.

$$C_n = \begin{pmatrix} H_{2^{d-1}} & \mathbf{0}_{2^{d-1} \times (n-2^{d-1})} \\ \mathbf{0}_{(n-2^{d-1}) \times 2^{d-1}} & \mathbf{0}_{(n-2^{d-1}) \times (n-2^{d-1})} \end{pmatrix},$$

where $H_{2^{d-1}}$ is the $2^{d-1} \times 2^{d-1}$ Hadamard matrix. We move the first 2^{d-1} rows of C_n to the rows $i_1, \ldots, i_{2^{d-1}}$, respectively. This process can be achieved through a sequence of exchanges of two rows. In other words, there is an $n \times n$ transformation matrix A such that AC_n does the work and $AA^T = n\mathbf{I}_n$. We now move the first 2^{d-1} columns of AC_n

to the columns $i_1, \ldots, i_{2^{d-1}}$, respectively. Similarly, this process can be achieved through a sequence of exchanges of two columns. Moreover, AC_nA^T does the work. Now, for any j with $1 \leq j \leq 2^{d-1}$, let $\mathbf{d}_{i_j} = (x_1, \ldots, x_n)$ denote the i_j-th row of AC_nA^T. Then, for $1 \leq s \leq 2^{d-1}$, the i_s-th component of $\mathbf{d}_{i_j}$ denoted by x_{i_s} is in fact the s-th component of the j-th row of the Hadamard matrix $H_{2^{d-1}}$. In other words, $\mathbf{d}_{i_j}$ has all zero components except these 2^{d-1} components x_{i_s} forming a subvector that is the same as the j-th row of $H_{2^{d-1}}$. We finally construct $\mathbf{v}'_j$ from $\mathbf{d}_{i_j}$ by changing all its -1 components to zero and keeping all its other components. We also construct $\mathbf{v}''_j$ from $\mathbf{d}_{i_j}$ by changing all its -1 components to one and all its one components to zero, and keeping all the other components. In other words, $\mathbf{v}'_j$ and $\mathbf{v}''_j$ are constructed as follows:

- We first construct $\mathbf{z}'_j$ and $\mathbf{z}''_j$ from the $2^{d-1} \times 2^{d-1}$ Hadamard matrix $H_{2^{d-1}}$ as in Definition 10.
- We construct $\mathbf{v}'_j$ by adding $n - 2^{d-1}$ zero components to the end of $\mathbf{z}'_j$ and then moving the first 2^{d-1} components of $\mathbf{z}'_j$ to $i_1, \ldots, i_{2^{d-1}}$ through exchanging columns.
- Similarly, we construct $\mathbf{v}''_j$ by adding $n - 2^{d-1}$ zero components to the end of $\mathbf{z}''_j$ and then moving the first 2^{d-1} components of $\mathbf{z}''_j$ to $i_1, \ldots, i_{2^{d-1}}$ through exchanging columns.

Hence, Proposition 11 follows from Proposition 8 in a manner similar to Proposition 10. □

In the following two lemmas we show that the sequence B_H enables the query vector $\mathbf{q}_t$ to preserve m_1 similarity for any pair of $\mathbf{z}'_t$ and $\mathbf{z}''_t$ in B_H when the zero initial query vector is used, and the sequence $D(\mathbf{q}_1)$ enables the query vector $\mathbf{q}_t$ to preserve m_1 similarity for any pair of $\mathbf{v}'_t$ and $\mathbf{v}''_t$ in $D(\mathbf{q}_1)$ when the arbitrary initial query vector $\mathbf{q}_1$ is used.

LEMMA 12. *For $n = 2^d + k - 1$ with positive integers k and d, let B_H be the sequence defined in Definition 9. Let*

$$\mathbf{q}_t = \alpha_{t_0}\mathbf{q}_1 + \sum_{j=1}^{t-1} \alpha_{t_j}\mathbf{x}_j$$

for $t = 1, \ldots, 2^d$, where the initial query vector $\mathbf{q}_1 = \mathbf{0}$, $\alpha_{t_j} \in R$, and $\mathbf{x}_j \in \{\mathbf{z}'_j, \mathbf{z}''_j\}$. Then, $m_1(\mathbf{q}_t, \mathbf{z}'_t) = m_1(\mathbf{q}_t, \mathbf{z}''_t)$ for $1 \leq t \leq 2^d$.

Proof. For any $1 \le j < t \le 2^d$, for $\mathbf{x}_j \in \{\mathbf{z}'_j, \mathbf{z}''_j\}$, by Proposition 10 (b) and (c) we have $m_1(\mathbf{x}_j, \mathbf{z}'_t) = m_1(\mathbf{x}_j, \mathbf{z}''_t)$. Hence, for any $1 < t \le 2^d$,

$$\begin{aligned} m_1(\mathbf{q}_t, \mathbf{z}'_t) &= m_1(\sum_{j=1}^{t-1} \alpha_{t_j} \mathbf{x}_j, \mathbf{z}'_t) = \sum_{j=1}^{t-1} \alpha_{t_j} m_1(\mathbf{x}_j, \mathbf{z}'_t) \\ m_1(\mathbf{q}_t, \mathbf{z}''_t) &= m_1(\sum_{j=1}^{t-1} \alpha_{t_j} \mathbf{x}_j, \mathbf{z}''_t) = \sum_{j=1}^{t-1} \alpha_{t_j} m_1(\mathbf{x}_j, \mathbf{z}''_t) \\ &= \sum_{j=1}^{t-1} \alpha_{t_j} m_1(\mathbf{x}_j, \mathbf{z}'_t) \end{aligned}$$

For $t = 1$, we have $m_1(\mathbf{q}_1, \mathbf{z}'_1) = m_1(\mathbf{q}_1, \mathbf{z}''_1) = 0$, since $\mathbf{q}_1 = \mathbf{0}$. Thus, for any $1 \le t \le 2^d$, $m_1(\mathbf{q}_t, \mathbf{z}'_t) = m_1(\mathbf{q}_t, \mathbf{z}''_t)$. □

LEMMA 13. *Let $n = 2^d + k - 1$ with positive integers k and d. Given any initial query vector $\mathbf{q}_1 \in \{0,1\}^n$, let $D(\mathbf{q}_1)$ be the sequence given in Proposition 11, and*

$$\mathbf{q}_t = \alpha_{t_0} \mathbf{q}_1 + \sum_{j=1}^{t-1} \alpha_{t_j} \mathbf{x}_j$$

for $t = 1, \ldots, 2^{d-1}$, where $\alpha_{t_j} \in R$ and $\mathbf{x}_j \in \{\mathbf{v}'_j, \mathbf{v}''_j\}$. Then, $m_1(\mathbf{q}_t, \mathbf{v}'_t) = m_1(\mathbf{q}_t, \mathbf{v}''_t)$ for $2 \le t \le 2^{d-1}$. Moreover, $m_1(\mathbf{q}_1, \mathbf{v}'_1) = m_1(\mathbf{q}_1, \mathbf{v}''_1)$ if $\mathbf{q}_1$ has at least 2^{d-1} zero components.

Proof. For any $1 \le j < t \le 2^{d-1}$, for $\mathbf{x}_j \in \{\mathbf{v}'_j, \mathbf{v}''_j\}$, by Proposition 11, properties similar to (b) and (c) of Proposition 10 hold for the sequence $D(\mathbf{q}_1)$. Hence, we have $m_1(\mathbf{x}_j, \mathbf{v}'_t) = m_1(\mathbf{x}_j, \mathbf{v}''_t)$. Thus, for any $2 \le t \le 2^{d-1}$, we have by Proposition 11

$$\begin{aligned} m_1(\mathbf{q}_t, \mathbf{v}'_t) &= m_1(\alpha_{t_0} \mathbf{q}_1 + \sum_{j=1}^{t-1} \alpha_{t_j} \mathbf{x}_j, \mathbf{v}'_t) \\ &= \alpha_{t_0} m_1(\mathbf{q}_1, \mathbf{v}'_t) + m_1(\sum_{j=1}^{t-1} \alpha_{t_j} \mathbf{x}_j, \mathbf{v}'_t) \\ &= \sum_{j=1}^{t-1} \alpha_{t_j} m_1(\mathbf{x}_j, \mathbf{v}'_t) \\ m_1(\mathbf{q}_t, \mathbf{v}''_t) &= m_1(\alpha_{t_0} \mathbf{q}_1 + \sum_{j=1}^{t-1} \alpha_{t_j} \mathbf{x}_j, \mathbf{v}''_t) \\ &= \alpha_{t_0} m_1(\mathbf{q}_1, \mathbf{v}''_t) + m_1(\sum_{j=1}^{t-1} \alpha_{t_j} \mathbf{x}_j, \mathbf{v}''_t) \end{aligned}$$

$$= \sum_{j=1}^{t-1} \alpha_{t_j} m_1(\mathbf{x}_j, \mathbf{v''}_t) = \sum_{j=1}^{t-1} \alpha_{t_j} m_1(\mathbf{x}_j, \mathbf{v'}_t)$$

Thus, for any $2 \leq t \leq 2^{d-1}$, $m_1(\mathbf{q}_t, \mathbf{v'}_t) = m_1(\mathbf{q}_t, \mathbf{v''}_t)$. If $\mathbf{q}_1$ has at least 2^{d-1} zero components, then again by Proposition 11 we have $m_1(\mathbf{q}_1, \mathbf{v'}_1) = m_1(\mathbf{q}_1, \mathbf{v''}_1) = 0$. □

The following lemma allows us to choose examples in some subdomain to force a learning algorithm to make mistakes.

LEMMA 14. *Given $n > k-1 \geq 0$, there is an adversary strategy that forces any adaptive supervised learning algorithm to make at least $k-1$ mistakes for learning the class of disjunctions of at most $k-1$ variables from $\{x_{i_1}, \ldots, x_{i_{k-1}}\}$ over the binary vector space $\{0,1\}^n$. Moreover, the adversary chooses examples in the vector space with nonzero values only for variables in $\{x_{i_1}, \ldots, x_{i_{k-1}}\}$.*

Proof. For any given adaptive supervised learning algorithm, at any step t for $1 \leq t \leq k-1$, the adversary uses the example $\mathbf{x}_t$ to defeat the learning algorithm as follows, where $\mathbf{x}_t$ has all zero components except that its i_t-th components is one: If the learning algorithm classifies $\mathbf{x}_t$ as relevant, then the adversary classifies it as irrelevant, otherwise the adversary classifies it as relevant. □

One may easily note that the strategy we used above to prove Lemma 14 can be generalized to prove the following fact: Any given adaptive supervised learning algorithm makes at least n mistakes to learn the class of disjunctions of at most n variables from $\{x_1, \ldots, x_n\}$ over the binary vector space $\{0,1\}^n$, because the algorithm can be forced to make one mistake to determine whether or not each of the n variables is in the target disjunction of at most n variables. Unfortunately, one must note that this kind of strategy cannot be used to prove that an adaptive supervised learning algorithm (such as Rocchio's algorithm) makes at least n mistakes for learning a disjunction of at most k variable when k is less than n, especially, when k is a small constant. That is the reason why we must design sophisticated methods in this paper to prove linear lower bounds for Rocchio's algorithm for learning disjunctions of at most k variables, where k can be any value between 1 and n. For example, k can be 1, 3, $\log n$, or $\frac{n}{10}$. Disjunctions of a small number of variables are the common ways for users to specify their information needs. For example, in the real-world of web search, the number of keywords used in a query session is usually very small. The problem of learning disjunctions of a small number of variables

has been studied by many researchers (see, for example, the work in (Littlestone, 1988)).

Careful readers may have observed that we do not rule out the choice of $d = 0$ in the decomposition of $n = 2^d + k - 1$ from Proposition 8 to Lemma 13 in this section. When $d = 0$, we have $k = n$, hence Lemma 14 can be simply used to prove linear lower bounds for any adaptive supervised learning algorithm (such as Rocchio's algorithm) to learn disjunctions of at most $k = n$ variables over the binary vector space. However, as we pointed out in the above paragraph, for any k such that $1 \leq k < n$, we must reply on the choice of $d \neq 0$ in the decomposition of $n = 2^d + k - 1$ to prove our linear lower bounds. For example, in the case of $k = 1$, with the choice of $d \neq 0$ such that $n = 2^d$ we can prove that Rocchio's algorithm makes at least $n = 2^d$ mistakes for learning disjunctions of *one* variable when the zero initial query vector is used. As we will show in the next section, the proof is accomplished through the construction of 2^d pairs of examples from row vectors of Hadamard matrix $\mathbf{H}_{2^d}$. Once again, the strategy of Lemma 14 or the like is not applicable at all to the case of learning disjunctions of *one* variable, nor to the case of learning disjunctions of a small number of variables.

4. Linear Lower Bounds

We are now ready to prove linear lower bounds for Rocchio's similarity-based relevance feedback algorithm when any of the four typical similarities is used. Throughout this section, we let $n = 2^d + k - 1$ with two positive integers d and k, and let $\mathbf{u}$ be the vector in $\{0,1\}^n$ such that its first component is one, its last $k-1$ components have all $k-1$ or fewer ones (however, these one-components are not specified at this point), and all other components are zero. Given any query vector $\mathbf{q}_1 \in \{0,1\}^n$, let $q_{i_1}, \ldots, q_{i_{2^{d-1}}}$ be its 2^{d-1} components such that they are either all zero or all one. Define $\mathbf{u}(\mathbf{q}_1)$ to be the vector in $\{0,1\}^n$ such that its i_1-th component is one, its i_j-th components are all zero for $j = 2, \ldots, 2^{d-1}$, and among the remaining $n - 2^{d-1}$ components there are at most $k-1$ one components (again, setting which of these components to be one is not determined at this point). Note that both $\mathbf{u}$ and $\mathbf{u}(\mathbf{q}_1)$ define respectively a monotone disjunction of at most k relevant features. We use $E(\mathbf{u})$ and $E(\mathbf{u}(\mathbf{q}_1))$ to denote the monotone disjunctions represented by $\mathbf{u}$ and $\mathbf{u}(\mathbf{q}_1)$, respectively.

LEMMA 15. *Let $B_H = ((\mathbf{z'}_1, \mathbf{z''}_1), \ldots, (\mathbf{z'}_{2^d}, \mathbf{z''}_{2^d})$ be the example sequence defined in Definition 9. For any similarity m_i, $1 \leq i \leq 4$, there*

is a $\psi \in R$ such that the query vector

$$\mathbf{q}_t = \alpha_{t_0}\mathbf{q}_1 + \sum_{j=1}^{t-1} \alpha_{t_j}\mathbf{x}_j$$

and the classifier $(\mathbf{u}, \psi, m_i)$ differ on B_H with respect to m_i for $t = 1, \ldots, 2^d$, where α_{t_j} are arbitrary values in R, and $\mathbf{x}_j \in \{\mathbf{z'}_j, \mathbf{z''}_j\}$.

Proof. As noted in section 2, $(\mathbf{u}, 1/2, m_1)$, $(\mathbf{u}, 2/(k+n), m_2)$, $(\mathbf{u}, 1/\sqrt{kn}, m_3)$, and $(\mathbf{u}, 1/(k+n-1), m_4)$ are respectively classifiers for the monotone disjunction $E(\mathbf{u})$ of at most k relevant features. It follows from Definition 9 that the first component of $\mathbf{z'}_t$ is one, the first of $\mathbf{z''}_t$ is zero, and the last k components of each of both $\mathbf{z'}_t$ and $\mathbf{z''}_t$ are all zero for $1 \le t \le 2^d$. Hence, for any $1 \le t \le 2^d$, we have by Proposition 10

$$\begin{aligned}
m_1(\mathbf{u}, \mathbf{z'}_t) &= 1 > \frac{1}{2}, && (8)\\
m_2(\mathbf{u}, \mathbf{z'}_t) &= \frac{2m_1(\mathbf{u}, \mathbf{z'}_t)}{m_1(\mathbf{u}, \mathbf{u}) + m_1(\mathbf{z'}_t, \mathbf{z'}_t)} \ge \frac{2}{k + 2^{d-1}} > \frac{2}{k+n}, && (9)\\
m_3(\mathbf{u}, \mathbf{z'}_t) &= \frac{m_1(\mathbf{u}, \mathbf{z'}_t)}{\sqrt{m_1(\mathbf{u}, \mathbf{u})}\sqrt{m_1(\mathbf{z'}_t, \mathbf{z'}_t)}} \ge \frac{1}{\sqrt{k2^{d-1}}} > \frac{1}{\sqrt{kn}}, && (10)\\
m_4(\mathbf{u}, \mathbf{z'}_t) &= \frac{m_1(\mathbf{u}, \mathbf{z'}_t)}{m_1(\mathbf{u}, \mathbf{u}) + m_1(\mathbf{z'}_t, \mathbf{z'}_t) - m_1(\mathbf{u}, \mathbf{z'}_t)}\\
&\ge \frac{1}{k + 2^{d-1} - 1} > \frac{1}{k+n-1}, \quad \textit{and} && (11)\\
m_i(\mathbf{u}, \mathbf{z''}_t) &= 0, \ \textit{for } 1 \le i \le 4. && (12)
\end{aligned}$$

By Lemma 12 and the above (8) and (12), for any $1 \le t \le 2^d$, $\mathbf{q}_t$ and the classifier $(\mathbf{u}, 1/2, m_1)$ differ on the sequence B_H with respect to m_1.

For the similarity m_2, for any $t \ge 2$ we have by Proposition 10 and Lemma 12

$$\begin{aligned}
m_2(\mathbf{q}_t, \mathbf{z'}_t) &= \frac{2m_1(\mathbf{q}_t, \mathbf{z'}_t)}{m_1(\mathbf{q}_t, \mathbf{q}_t) + m_1(\mathbf{z'}_t, \mathbf{z'}_t)} = \frac{2m_1(\mathbf{q}_t, \mathbf{z'}_t)}{m_1(\mathbf{q}_t, \mathbf{q}_t) + 2^{d-1}},\\
m_2(\mathbf{q}_t, \mathbf{z''}_t) &= \frac{2m_1(\mathbf{q}_t, \mathbf{z''}_t)}{m_1(\mathbf{q}_t, \mathbf{q}_t) + m_1(\mathbf{z''}_t, \mathbf{z''}_t)} = \frac{2m_1(\mathbf{q}_t, \mathbf{z'}_t)}{m_1(\mathbf{q}_t, \mathbf{q}_t) + 2^{d-1}}\\
&= m_2(\mathbf{q}_t, \mathbf{z'}_t),
\end{aligned}$$

For $t = 1$, we have $m_1(\mathbf{q}_1, \mathbf{z'}_1) = m_1(\mathbf{0}, \mathbf{z'}_1) = 0$, hence $m_2(\mathbf{q}_1, \mathbf{z'}_1) = 0$. Because $\mathbf{z''}_1 = \mathbf{q}_1 = \mathbf{0}$, we also have $m_2(\mathbf{q}_1, \mathbf{z''}_1) = 0$ according to the definition. Thus, for $1 \le t \le 2^d$, we have $m_2(\mathbf{q}_t, \mathbf{z'}_t) = m_2(\mathbf{q}_t, \mathbf{z''}_t)$, hence by (9) and (12) the query vector $\mathbf{q}_t$ and the classifier $(\mathbf{u}, 2/(k+n), m_2)$ differ on the sequence B_H with respect to m_2.

For the similarity m_3, for any $t \geq 2$ we have by Proposition 10 and Lemma 12

$$\begin{aligned} m_3(\mathbf{q}_t, \mathbf{z'}_t) &= \frac{m_1(\mathbf{q}_t, \mathbf{z'}_t)}{\sqrt{m_1(\mathbf{q}_t, \mathbf{q}_t)}\sqrt{m_1(\mathbf{z'}_t, \mathbf{z'}_t)}} = \frac{m_1(\mathbf{q}_t, \mathbf{z'}_t)}{\sqrt{m_1(\mathbf{q}_t, \mathbf{q}_t)}\sqrt{2^{d-1}}}, \\ m_3(\mathbf{q}_t, \mathbf{z''}_t) &= \frac{m_1(\mathbf{q}_t, \mathbf{z''}_t)}{\sqrt{m_1(\mathbf{q}_t, \mathbf{q}_t)}\sqrt{m_1(\mathbf{z''}_t, \mathbf{z''}_t)}} = \frac{m_1(\mathbf{q}_t, \mathbf{z'}_t)}{\sqrt{m_1(\mathbf{q}_t, \mathbf{q}_t)}\sqrt{2^{d-1}}} \\ &= m_3(\mathbf{q}_t, \mathbf{z'}_t), \end{aligned}$$

For $t = 1$, as for the similarity m_1, $m_3(\mathbf{q}_1, \mathbf{z'}_1) = 0$. Because $\mathbf{z''}_1 = \mathbf{q}_1 = \mathbf{0}$, we also have $m_3(\mathbf{q}_1, \mathbf{z''}_1) = 0$ according to the definition. Thus, for $1 \leq t \leq 2^d$, we have $m_3(\mathbf{q}_t, \mathbf{z'}_t) = m_3(\mathbf{q}_t, \mathbf{z''}_t)$, hence by (10) and (12) the query vector $\mathbf{q}_t$ and the classifier $(\mathbf{u}, 1/\sqrt{kn}, m_3)$ differ on the sequence B_H with respect to m_3.

Finally, for the similarity m_4, for any $t \geq 2$ we have by Proposition 10 and Lemma 12

$$\begin{aligned} m_4(\mathbf{q}_t, \mathbf{z'}_t) &= \frac{m_1(\mathbf{q}_t, \mathbf{z'}_t)}{m_1(\mathbf{q}_t, \mathbf{q}_t) + m_1(\mathbf{z'}_t, \mathbf{z'}_t) - m_1(\mathbf{q}_t, \mathbf{z'}_t)} \\ &= \frac{m_1(\mathbf{q}_t, \mathbf{z'}_t)}{m_1(\mathbf{q}_t, \mathbf{q}_t) + 2^{d-1} - m_1(\mathbf{q}_t, \mathbf{z'}_t)}, \\ m_4(\mathbf{q}_t, \mathbf{z''}_t) &= \frac{m_1(\mathbf{q}_t, \mathbf{z''}_t)}{m_1(\mathbf{q}_t, \mathbf{q}_t) + m_1(\mathbf{z''}_t, \mathbf{z''}_t) - m_1(\mathbf{q}_t, \mathbf{z''}_t)} \\ &= \frac{m_1(\mathbf{q}_t, \mathbf{z'}_t)}{m_1(\mathbf{q}_t, \mathbf{q}_t) + 2^{d-1} - m_1(\mathbf{q}_t, \mathbf{z'}_t)} \\ &= m_4(\mathbf{q}_t, \mathbf{z'}_t), \end{aligned}$$

For $t = 1$, as for m_2 and m_3 we have $m_4(\mathbf{q}_1, \mathbf{z'}_1) = m_4(\mathbf{q}_1, \mathbf{z''}_1) = 0$. Thus, for $1 \leq t \leq 2^d$, we have $m_4(\mathbf{q}_t, \mathbf{z'}_t) = m_4(\mathbf{q}_t, \mathbf{z''}_t)$, hence by (11) and (12) the query vector $\mathbf{q}_t$ and the classifier $(\mathbf{u}, 1/(k+n-1), m_4)$ differ on the sequence B_H with respect to m_4. □

LEMMA 16. *Given any initial query vector $\mathbf{q}_1 \in \{0,1\}^n$, let $D(\mathbf{q}_1) = ((\mathbf{v'}_1, \mathbf{v''}_1), \ldots, (\mathbf{v'}_{2^{d-1}}, \mathbf{v''}_{2^{d-1}})$ be the example sequence defined in Proposition 11. For any similarity m_i, $1 \leq i \leq 4$, there is a $\psi \in R$ such that the query vector*

$$\mathbf{q}_t = \alpha_{t_0}\mathbf{q}_1 + \sum_{j=1}^{t-1} \alpha_{t_j}\mathbf{x}_j$$

and the classifier $(\mathbf{u}, \psi, m_i)$ differ on $D(\mathbf{q}_1)$ with respect to m_i for $t = 2, \ldots, 2^{d-1}$, where α_{t_j} are arbitrary values in R, and $\mathbf{x}_j \in \{\mathbf{v'}_j, \mathbf{v''}_j\}$. Moreover, if $\mathbf{q}_1$ has at least 2^{d-1} zero components, then $\mathbf{q}_1$ and $(\mathbf{u}, \psi, m_i)$ differ with respect to m_i, too.

Proof. The proof is the same as what we just did for Lemma 15, but we need to replace $\mathbf{u}$ by $\mathbf{u}(\mathbf{q}_1)$, $\mathbf{z}$ by $\mathbf{v}$, and 2^d by 2^{d-1}, respectively. We also need to use Proposition 11 and Lemma 13 to complete our proof. □

We now prove the following main results in this paper.

THEOREM 17. *Let $n = 2^d + k - 1$ for some positive integers d and k. For any given similarity m_i with $i \in \{1, 2, 3, 4\}$, Rocchio's similarity-based relevance feedback algorithm makes at least n mistakes for learning the class of monotone disjunctions of at most k relevant features over the binary vector space $\{0, 1\}^n$, when the initial query vector $\mathbf{q}_1 = \mathbf{0}$ and the similarity m_i are used.*

Proof. Let A be the Rocchio's similarity-based relevance feedback algorithm with the similarity m_i and the initial query vector $\mathbf{q}_1 = \mathbf{0}$. We analyze the number of mistakes that A must make in learning $E(\mathbf{u})$, the disjunction of at most k relevant features represented by $\mathbf{u}$. As we noted before, there is a $\psi \in R$ such that the classifier $(\mathbf{u}, \psi, m_i)$ classifies $E(\mathbf{u})$, i.e., it is logically equivalent to $E(\mathbf{u})$. By Lemma 15, the classifier $(\mathbf{u}, \psi, m_i)$ and the query vector $\mathbf{q}_t$ with $1 \leq t \leq 2^d$ differ on the example sequence $B_H = ((\mathbf{z'}_1, \mathbf{z''}_1), \ldots, (\mathbf{z'}_{2^d}, \mathbf{z''}_{2^d})$ with respect to the similarity m_i. Thus, by Lemma 6, the adversary can use examples from B_H to force the algorithm A to make 2^d mistakes. Note that the last $k - 1$ components of all examples in B_H are zero and the last $k - 1$ components of $\mathbf{u}$ are unspecified but may have at most $k - 1$ one components. Hence, by Lemma 14, the adversary can further force A to make at least $k - 1$ mistakes to learn the values of the last $k - 1$ components of $\mathbf{u}$. Putting all together, A makes at least $2^d + k - 1 = n$ mistakes for learning $E(\mathbf{u})$. □

THEOREM 18. Let $n = 2^d + k - 1$ for some positive integers d and k. For any given similarity m_i with $i \in \{1, 2, 3, 4\}$, Rocchio's similarity-based relevance feedback algorithm makes at least $(n+k-3)/2$ mistakes for learning the class of monotone disjunctions of at most k relevant features over the binary vector space $\{0, 1\}^n$, when an arbitrary initial query vector $\mathbf{q}_1 \in \{0, 1\}^n$ and the similarity m_i are used. Moreover, if the initial query vector $\mathbf{q}_1$ has at least 2^{d-1} zero components, then the algorithm makes at least $(n + k - 1)/2$ mistakes.

Proof. Let A be the Rocchio's similarity-based relevance feedback algorithm with the similarity m_i and the arbitrary initial query vector $\mathbf{q}_1 \in \{0, 1\}^n$. We analyze the number of mistakes that A must make in learning $E(\mathbf{u}(\mathbf{q}_1))$, the disjunction of at most k relevant features represented by $\mathbf{u}(\mathbf{q}_1)$. As we noted before, there is a $\psi \in R$ such that the

classifier $(\mathbf{u}(\mathbf{q}_1), \psi, m_i)$ classifies $E(\mathbf{u}(\mathbf{q}_1))$, i.e., it is logically equivalent to $E(\mathbf{u}(\mathbf{q}_1))$. By Lemma 16, the classifier $(\mathbf{u}(\mathbf{q}_1), \psi, m_i)$ and the query vector $\mathbf{q}_t$ with $2 \leq t \leq 2^d$ differ on the example sequence $D(\mathbf{q}_1) = ((\mathbf{v'}_1, \mathbf{v''}_1), \ldots, (\mathbf{v'}_{2^{d-1}}, \mathbf{v''}_{2^{d-1}})$ with respect to the similarity m_i. If $\mathbf{q}_1$ has at least 2^{d-1} zero components, then the classifier and $\mathbf{q}_1$ also differ with respect to m_i. Thus, by Lemma 6, the adversary can use examples from $D(\mathbf{q}_1)$ to force the algorithm A to make $2^{d-1}-1$ mistakes, or 2^{d-1} mistakes if $\mathbf{q}_1$ has at least 2^{d-1} zero components. Note that there are $n - 2^{d-1} > k - 1$ positions such that the components of all examples in $D(\mathbf{q}_1)$ at those positions are zero, and the components of $\mathbf{u}$ at those components are unspecified but may have at most $k-1$ one components. Hence, by Lemma 14, the adversary can further force A to make at least $k-1$ mistakes to learn the values of the components of $\mathbf{u}$ at those $n-2^{d-1}$ positions. Putting all together, A makes at least $2^{d-1}+k-2 = (n+k-3)/2$ mistakes for learning $E(\mathbf{u})$; and if $\mathbf{q}_1$ has at least 2^{d-1} zero components, A makes at least $2^{d-1}+k-1 = (n+k-1)/2$ mistakes. □

The lower bounds obtained in Theorems 17 and 18 are independent of the choices of the threshold and coefficients that Rocchio's similarity-based relevance feedback algorithm may use in updating its query vector and in making its classification.

5. Discussions

5.1. The Gradient Descent Procedure

As pointed out in (Wong et al., 1988), one primary concern in information retrieval is to ensure that those documents more relevant to the user information needs are ranked ahead of those less relevant ones. This means that a ranking is acceptable if it can guarantee that less preferred documents will not be listed in front of the more preferred ones (such a ranking is called an acceptable ranking in (Wong et al., 1988)). Let $\prec$ denote the user preference relation over the document vector space. Wong, Yao and Bollmann designed a very nice gradient descent procedure to compute the query vector q satisfying

$$d \prec d' \Longrightarrow m_1(q, b) > 0,$$

where $b = d' - d$ is called the *difference* vector for documents d and d'.

Gradient Descent Procedure. The procedure is outlined as follows:

(i) Choose an initial query vector q_0 and let $k = 0$.

(ii) Let q_k be the query vector in the k-th step. Identify the set of difference vectors

$$\Gamma(q_k) = \{b = d' - d | d \prec d',\ m_1(q, b) \leq 0\}.$$

If $\Gamma(q_k) = \emptyset$ (i.e., q_k is a solution vector), terminate the procedure.

(iii) Let

$$q_{k+1} = q_k + \sum_{b \in \Gamma(q_k)} b.$$

(iv) Let $k = k + 1$; go back to step (ii).

The above gradient descent procedure is a very nice adaptive algorithm for computing the query vector. However, it must know the user preference relation $\prec$ ahead of the time and perform exhaustive search to identify the set $\Gamma(q_k)$ at step (ii). The exhaustive search is of exponential time complexity. In practice and in a machine learning setting, the user preference relation $\prec$ is the unknown target that must be learned by an information retrieval system. In this sense, the gradient descent procedure is not an adaptive learning process. On the other hand, Rocchio's similarity-based relevance feedback algorithm formally defined in section 2 is an adaptive learning algorithm without a priori knowledge of the user preference. The updating process for the query vector at each iteration is of linear time complexity.

5.2. Learning Other Document Classes

In this paper, we study the learning of a document class represented by a monotone disjunction of index features (or terms) with Rocchio's algorithm. In fact, Rocchio's algorithm can be easily used to learn other target document classes such as the class represented by a conjunction of disjunctions of index features (or terms). Conjunctions of disjunctions are the most common forms for search queries as constructed by humans. We give several examples here. From those examples we know that the lower bounds proved in previous section on the learning performance of Rocchio's algorithms hold for those learning cases.

Example 1 (Learning Arbitrary Disjunctions). An arbitrary disjunction is a general case of monotone disjunctions and may have negated index features in it. A negated index feature in a disjunction means that the feature should not occur in the desired documents. Let

$$g(x_1, \ldots, x_n) = x_{i_1} \vee \cdots \vee x_{i_s} \vee \bar{x}_{j_1} \vee \cdots \vee \bar{x}_{j_t}$$

be an arbitrary disjunction. Let $z = (z_1, \ldots, z_n)$ be a document vector that makes g false, i.e., $g(z_1, \ldots, z_n) = 0$. Then, we must have $z_{i_1} =$

$\cdots = z_{i_s} = 0$ and $z_{j_1} = \cdots = z_{j_t} = 1$. We can use z to transform g into a monotone disjunction

$$yf(x_1, \ldots, x_n) = g(x_1 + z_1, \ldots, x_n + z_n) = x_{i_1} \vee \cdots \vee x_{i_s} \vee x_{j_1} \vee \cdots \vee x_{j_t}.$$

Please also note that the same z can be used to transform f back into g in the following manner

$$g(x_1, \ldots, x_n) = f(x_1 + z_1, \ldots, x_n + z_n).$$

The above transformation methods tell us that g can be learned as follows. First, use Rocchio's algorithm with an initial query vector q to learn g. When a document vector $z = (z_1, \ldots, z_n)$ is judged by the user as irrelevant, then one can use this z to transform any obtained document vector $d = (d_1, \ldots, d_n)$ into $d + z = (d_1 + z_1, \ldots, d_n + z_n)$ to continue the learning. With this kind of transformation, the algorithm actually learns the function f. But as we observed before, f can be easily transformed into g with the document vector z. The above process implies that learning an arbitrary disjunction with Rocchio's algorithm has the same performance as learning a monotone disjunction.

Example 2 (Learning Monotone Conjunctions). Let

$$g(x_1, \ldots, x_n) = x_{i_1} \wedge \cdots \wedge x_{i_s}$$

be a monotone conjunction, i.e., all index features occurred in g are positive. The negation of g is a disjunction

$$\bar{g}(x_1, \ldots, x_n) = \bar{x}_{i_1} \vee \cdots \vee \bar{x}_{i_s}.$$

Note that the vector $z = (1, \ldots, 1)$ can be used to transform $\bar{g}$ into a monotone disjunction

$$f(x_1, \ldots, x_n) = \bar{g}(x_1 + z_1, \ldots, x_n + z_n) = x_{i_1} \vee \cdots \vee x_{i_s},$$

and to transform f back into $\bar{g}$

$$\bar{g}(x_1, \ldots, x_n) = f(x_1 + z_1, \ldots, x_n + z_n).$$

Hence, to learn g we only need to learn its negation $\bar{g}$, and by example 1 this can be done with Rocchio's algorithm.

Example 3 (Learning Arbitrary Conjunctions). Let

$$g(x_1, \ldots, x_n) = x_{i_1} \wedge \cdots \wedge x_{i_s} \wedge \bar{x}_{j_1} \wedge \cdots \wedge \bar{x}_{j_t}$$

be an arbitrary conjunction. The negation $\bar{g}$ of g is

$$\bar{g}(x_1, \ldots, x_n) = \bar{x}_{i_1} \vee \cdots \vee \bar{x}_{i_s} \vee x_{j_1} \vee \cdots \vee x_{j_t}$$

which is an arbitrary disjunction. To learn g we only need to learn its negation $\bar{g}$, and by Example 1 this can be done with Rocchio's algorithm.

Example 4 (Learning Disjunctions of Conjunctions). Let

$$G(x_1, \ldots, x_n) = G_1(x_1, \ldots, x_n) \vee \cdots \vee G_s(x_1, \ldots, x_n)$$

be a disjunction of conjunctions, G_i is a conjunction for $1 \leq i \leq s$. Using the *virtual variable* technique developed in (Maass and Warmuth, 1998), one can learn G as follows: Introduce one new input variable (*virtual variable*) for each of the possible 3^n conjunctions that can be constructed from variables $x_1, \ldots, x_n$. When the values of $x_1, \ldots, x_n$ are known, then the value of each virtual variable is easy to be determined. With the help of virtual variables, G is a monotone disjunction and thus can be learned with Rocchio's algorithm.

Example 5 (Learning Conjunctions of Disjunctions). Let

$$F(x_1, \ldots, x_n) = F_1(x_1, \ldots, x_n) \wedge \cdots \wedge F_s(x_1, \ldots, x_n)$$

be a conjunction of disjunctions, F_i is a disjunction for $1 \leq i \leq s$. Because the negation of F is a disjunctions of conjunctions, as in Example 4 the *virtual variables* technique developed in (Maass and Warmuth, 1998) can be used to learn the negation of F and hence F itself.

5.3. A Remark on the Optimal Criterion of Rocchio's Algorithm

According to (J.J. Rocchio, 1971; Salton, 1989), the construction of an ideal query vector would target to maximize the average query-document similarity for the relevant documents and at the same time minimize the average of the query-document similarity for the irrelevant documents. It is known in (J.J. Rocchio, 1971; Salton, 1989) that under appropriate assumptions such an ideal query vector has the form

$$Q_{opt} = k \left\{ \frac{1}{R} \sum_{Rel} \frac{D_i}{|D_i|} - \frac{1}{N-R} \sum_{Nonrel} \frac{D_i}{|D_i|} \right\},$$

where R and $N - R$ are the assumed number of relevant and irrelevant documents, and the summations range over the sets of normalized relevant and irrelevant documents, respectively. However, the optimal query vector cannot be adopted in practice, because the sets of relevant and irrelevant documents with respect to the queries are not known

before an exhaustive search. Recall that the gradient descent procedure in (Wong et al., 1988) has a similar problem. On the other hand, the optimal query vector can be adaptively approached in the following (see (Salton, 1989)):

$$Q^{(i+1)} = Q^{(i)} + \frac{1}{|R'|}\sum_{D_i \in R'} D_i - \frac{1}{|N'|}\sum_{D_i \in N'} D_i, \ or$$

$$Q^{(i+1)} = Q^{(i)} + \alpha \sum_{D_i \in R'} D_i - \beta \sum_{D_i \in N'} D_i,$$

where R' is the set of documents judged by the user as relevant by the end of iteration i, and N' is the set of documents judged as irrelevant. In the above two approximation formulas, $\frac{1}{|R'|}$, $\frac{1}{|N'|}$, α and β are updating factors (or coefficients). In our formalization of Rocchio's algorithm given in section 2, arbitrary updating factors are allowed. The lower bounds we have proved are in fact independent of the choices of the updating factors. That is, our lower bounds hold even if $\frac{1}{|R'|}$ and $\frac{1}{|N'|}$ are used as updating factors.

5.4. Counting Arguments

For any k with $1 \leq k \leq n$, given any disjunction of k variables

$$Q = x_{i_1} \vee \cdots \vee x_{i_k},$$

it is easy to know that there are 2^{n-k} documents in the binary vector space $\{0,1\}^n$ that are irrelevant to Q, i.e., the vectors of those documents make Q false; and that there are $2^n - 2^{n-k}$ documents that are relevant to Q, i.e., the vectors of those documents make Q true. When k is small, say $k = 3$, 2^{n-3} is a huge value for a very large n. That is, there are a huge number of documents that are irrelevant to Q. One might ask whether or not this kind of observation helps us to find some simple ways to prove linear lower bounds for Rocchio's algorithm. For example, an adversary could, in response to almost any formulated query, produce some document which matches the query to some degree but it not relevant. As a matter of fact, this sort of idea is in essence an example of the decision tree technique developed in (Littlestone, 1988) to prove lower bounds for general learning algorithms:

- In the decision tree, each inner node is labeled with a document vector in the binary vector space $\{0,1\}^n$.
- There are two edges leaving each inner node, labeled "relevant" and "irrelevant", respectively.

- Each leaf is labeled with a disjunction of at most k variables in such a way that the disjunction at the leaf is consistent with all the labels along the path leading from the root to the leaf (the number of inner nodes along this path is the depth of the leaf).
- Assume that the depth of every leaf of the decision tree is at least t. Then, the mistake bound of any learning algorithm for learning disjunctions of at most k variables is at least t. Readers can refer (Littlestone, 1988) for the proof of this statement.

The decision tree technique is useful because it reduces the problem of proving a lower bound for every learning algorithm to the problem of constructing a single decision tree with the required depth. The latter can usually be done through counting arguments to estimate the depth of the decision tree. To our best knowledge, (Maass and Turán, 1994) is the best article to prove lower bounds for learning a threshold gate, a general case of a disjunction of at most k variables. The article presents several powerful counting arguments to estimate the depth of decision tree for the class of threshold gates. Unfortunately, those counting arguments do not help us to obtain linear lower bounds for Rocchio's algorithm. The reason is that there are *not many* monotone disjunctions of at most k binary variables. One can easily find out that the number of monotone disjunctions of at most k binary variables is

$$C(k) = \sum_{i=1}^{k} \left(\begin{array}{c} n \\ i \end{array} \right).$$

The least depth of the decision tree for the class of monotone disjunctions of at most k binary variables is $\log_2 C(k)$. For example,

$$\begin{aligned}
\log_2 C(1) &= \log_2 n = o(n), \\
\log_2 C(2) &= \log_2 \frac{n^2+n}{2} = o(n), \\
\log_2 C(n) &= \log_2 \sum_{i=1}^{n} \left(\begin{array}{c} n \\ i \end{array} \right) = \log_2(2^n - 1) < n, \\
\log_2 C(k) &< \log_2 C(n) < n, for\ any\ k < n.
\end{aligned}$$

Therefore, counting arguments cannot yield linear lower bounds for Rocchio's algorithm when k is much less than n, especially when k is a small constant. For example, when $k = 1$, the lower bound produced by the counting argument is just $\log_2 n$. However, the non-counting technique that we use is this paper can yield linear lower bounds for Rocchio's' algorithm for any k with $1 \leq k \leq n$, even if $k = 1$. Please recall that disjunctions of a small number of variables are the common

ways for users to specify their information needs. For example, in the real-world of web search, the number of keywords used in a query session is usually very small. The problem of learning disjunctions of a small number of variables has been studied by many researchers (see, for example, the work in (Littlestone, 1988)).

6. Conclusions and Open Problems

Rocchio's similarity-based relevance feedback algorithm is one of the most popular query reformation method in information retrieval and has been used in various applications. It is essentially an adaptive supervised learning algorithm from examples. However, there is little rigorous analysis of its learning complexity. In this paper we prove linear lower bounds for Rocchio's similarity-based relevance feedback algorithm when any of the four typical similarities listed in (Salton, 1989) is used. Because the linear lower bounds are proved with the worst case analysis, they may not affect the algorithm's effective applicability to the real-world problems. The lower bounds help us understand the nature of the algorithm well so that we may find new strategies to improve the effectiveness of Rocchio's algorithm or design new algorithms for information retrieval. One possible way is to use the Winnow2 (Littlestone, 1988) algorithm as an alternative for the similarity-based relevance feedback algorithm in applications of information retrieval. For example, in our recent research on building real-time intelligent web search engines (Chen et al., 2000; Chen and Meng, 2000), we used a tailored version of Winnow2.

We list the following open problems for future research.

PROBLEM 1. The lower bound in Theorem 18 holds for an arbitrary initial query vector $\mathbf{q}_1 \in \{0,1\}^n$. Choosing a zero-one initial query vector is a very common practice in applications. For example, in web search an initial zero-one query vector may be constructed with the query words submitted by the user. When the initial query vector $\mathbf{q}_1$ is chosen from R^n, we can prove the same lower bound with the similar but tedious approach for the similarities m_1, m_2, and m_3. But we do not know whether the same lower bound still holds for m_4.

PROBLEM 2. In this paper we have proved linear lower bounds for Rocchio's similarity-based relevance feedback algorithm in the binary vector space. We do not know whether our approach can be extended to study the learning complexity of Rocchio's algorithm in arbitrary discretized vector space.

PROBLEM 3. The linear lower bounds we have established for Rocchio's algorithm in the binary vector is based on the worst-case anal-

ysis. It would be very interesting to analyze the average-case learning complexity of Rocchio's algorithm. We feel that this problem is very challenging, because any nontrivial average case analysis will reply on realistic models of document distribution, index term distribution, and the user preference distributions as well. We feel that it is not easy to model those distributions nor to analyze the complexity under those distributions. The probabilistic corpus model proposed in (Papadimitrious et al., 2000) may shed some light on this problem.

PROBLEM 4. Although it follows from our linear lower bounds that the Winnow2 algorithm (Littlestone, 1988) has better worst case learning complexity, the authors do not know any provable theoretical analysis of the average learning complexity about the Winnow2 algorithm, nor about the similarity-base relevance feedback algorithm. More precisely, we do not know whether the Winnow2 algorithm performs better *in average* than the Rocchio's similarity-based relevance feedback algorithm.

Acknowledgements

In early 1997, Dr. Stanley Sclaroff asked the first author whether the relevance feedback algorithm can be used to help the user search for the desired World Wide Web documents with about *two dozens of examples* judged by the user. At the time, his research group implemented ImageRover (Taycher et al., 1997; Sclaroff et al., 1997), an image search engine from the user's relevance feedback, while the author and his colleagues started to build intelligent search tools (such as Yarrow (Chen and Meng, 2000), WebSail (Chen et al., 2000) and Features (Chen et al., 2001)) with the help of information retrieval and machine learning techniques. Dr. Sclaroff's question together with the authors' own research on building intelligent web search tools inspired the work in this paper. The authors would also acknowledge that the example sequence selection method developed in (Kivinen et al., 1997) for proving linear lower bounds for the Perceptron algorithm is the key to the breakthrough of our proofs. Without knowing the method it would take longer for the authors to finish the work in this paper.

We thank three anonymous referees and the journal editor, Professor Paul B. Kantor, for their critical and valuable questions and comments for helping us to revise this paper. We thank one referee for informing us the reference (Wong et al., 1988).

References

Angluin D (1987) Queries and concept learning. *Machine Learning* **2**(4): 319–432.

Baeza-Yates R and Ribeiro-Neto B (1999), eds. Modern Information Retrieval. Addison-Wesley.

Chen Z (2001) Multiplicative adaptive algorithms for user preference retrieval. In: Proceedings of the Seventh Annual International Computing and Combinatorics Conference. Springer-Verlag, 2001.

Chen Z and Meng X (2000) Yarrow: A real-time client site meta search learner. In: Proceedings of the AAAI 2000 Workshop on Artificial Intelligence for Web Search. AAAI Press, 2000. pp. 12–17,

Chen Z, Meng X, Fowler R and Zhu B (2001) FEATURES: Real-time adaptive feature learning and document learning. Journal of the American Society for Information Science **52**(8):655-665.

Chen Z, Meng X, Zhu B and Fowler R (2000) WebSail: From on-line learning to web search. In: Q. Li and et al. (eds.): Proceedings of the 2000 International Conference on Web Information Systems Engineering (the full version will appear in Journal of Knowledge and Information Science, the special issue of WISE'00). IEEE Press, 2000. pp. 192–199.

Frakes W and Baeza-Yates R (1992), eds. Information Retrieval: Data Structures and Algorithms. Prentice Hall.

Ide E. (1971a) Interactive search strategies and dynamic file organization in information retrieval. In: Salton G, ed. The Smart System - Experiments in Automatic Document Processing. Prentice-Hall, Englewood Cliffs, NJ, 1971. pp. 373–393.

Ide E (1971b) New experiments in relevance feedback. In: Salton G, ed. The Smart System - Experiments in Automatic Document Processing. Prentice-Hall, Englewood Cliffs, NJ, 1971. pp. 337–354.

Rocchio J (1971) Relevance feedback in information retrieval. In: Salton G, ed. The Smart Retrieval System - Experiments in Automatic Document Processing. Prentice-Hall, Englewood Cliffs, NJ, 1971. pp. 313–323.

Kivinen J, Warmuth M and Auer P (1997) The perceptron algorithm vs. Winnow: linear vs. logarithmic mistake bounds when few input variables are relevant. Artificial Intelligence, (1-2):325–343.

Lewis D (1991) Learning in intelligent information retrieval. In: Proceedings of the Eighth International Workshop on Machine Learning. pp. 235–239.

Littlestone N (1988) Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. Machine Learning **2**:285–318.

Maass W and Turán G (1994) How fast can a threshold gate learn?. Computational Learning Theory and Natural Learning Systems **1**:381–414.

Maass W and Warmuth M (1998) Efficient Learning with virtual threshold gates. Information and Computation **141**(1):66–83.

Papadimitriou C, Raghavan P and Tamaki H (2000) Latent semantic indexing: A probabilistic analysis. Journal of Computer and System Science **61**(2):217–235.

Raghavan V and Wong S (1986) A critical analysis of the vector space model for information retrieval. Journal of the American Society for Information Science **37**(5):279–287.

Rosenblatt F (1958) The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review **65**(6):386–407.

Salton G (1989), ed. Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley.

Salton G and Buckley C (1990) Improving retrieval performance by relevance feedback. Journal of the American Society for Information Science **41**(4):288–297.

Salton G, Wong S and Yang C (1975) A vector space model for automatic indexing. Comm. of ACM **18**(11):613–620.

Sclaroff S, Taycher L, and Cascia M (1997) ImageRover: A content-based image browser for the World Wide Web. In: Proceedings of the IEEE Workshop on Content-based Access of Image and Video Libraries. IEEE Press, 1997. pp. 2-9.

Taycher L., Cascia M and Sclaroff S (1997) Image digestion and relevance feedback in the ImageRover WWW search engines. In: Proceedings of the International Conference on Visual Information. pp. 85–92.

Wong S, Yao Y and Bollmann P (1988) Linear structures in information retrieval. In: Proceedings of the 1988 ACM-SIGIR Conference on Information Retrieval. pp. 219–232.