Training Algorithms for Linear Text Classifiers

David D. Lewis   Robert E. Schapire
AT&T Laboratories
Murray Hill, NJ 07974 USA
{*lewis, schapire*}*@research.att.com*

James P. Callan   Ron Papka
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA 01003 USA
{*callan, papka*}*@cs.umass.edu*

Abstract

Systems for text retrieval, routing, categorization and other IR tasks rely heavily on linear classifiers. We propose that two machine learning algorithms, the Widrow-Hoff and EG algorithms, be used in training linear text classifiers. In contrast to most IR methods, theoretical analysis provides performance guarantees and guidance on parameter settings for these algorithms. Experimental data is presented showing Widrow-Hoff and EG to be more effective than the widely used Rocchio algorithm on several categorization and routing tasks.

1 Introduction

Document retrieval, categorization, routing, and filtering systems often are based on classification. That is, the IR system decides for each document which of two or more classes it belongs to, or how strongly it belongs to a class, in order to accomplish the IR task of interest. For instance, the two classes may be the documents relevant and not relevant to a particular user, and the system may rank documents based on how likely it is that they belong to the relevance class.

The rules or *classifiers* used to perform these tasks are often trained on data rather than, or subsequent to, being constructed by hand. For instance, a ranked retrieval system using relevance feedback will ask its user to indicate which of the top ranked documents retrieved for a query are relevant and which are not. The judged documents are used as training data to produce a more effective query and thus a new and better ranking. In text categorization, documents that have been categorized by human indexers can be used as training data for a classifier to categorize future documents.

In this paper we compare the widely used Rocchio training algorithm to two other algorithms, the Widrow-Hoff and EG algorithms, which produce the same kind of classifier, a linear classifier. The Widrow-Hoff and EG algorithms are better understood from a theoretical standpoint, leading to performance guarantees and guidance on parameter settings. In addition, we show experimentally that Widrow-Hoff and EG are more effective than Rocchio on both routing and categorization tasks.

Permission to make digital/hard copy of all or part of this material without fee is granted provided that copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery, Inc. (ACM). To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

2 Linear Functions in IR

IR systems often represent texts as *feature vectors*, that is, tuples of values:

$$\mathbf{x} = (x_1, x_2, \ldots, x_d)$$

where x_j is the numeric value that feature j takes on for this document, and d is the number of features. For example, d might be the number of distinct non-stopwords in a textbase, and x_j the number of times a particular word occurs in this document.

In order to rank documents, a text retrieval system typically applies a d-ary function f to each vector $\mathbf{x}$, producing a score $f(\mathbf{x})$. Documents with the largest values of $f(\mathbf{x})$ appear at the top of a ranking. A text categorization system might similarly compute scores $f(\mathbf{x})$ and assign to a category only those documents where $f(\mathbf{x})$ exceeds some threshold or satisfies some other criterion. Systems for filtering, routing, and other text classification tasks operate similarly.

The simplest such functions are *linear*, that is, they may be expressed as the dot product of a weight vector $\mathbf{w}$ and the feature vector $\mathbf{x}$:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} = \sum_{j=1}^{d} w_j x_j.$$

Most approaches to ranked retrieval use linear functions. For instance, in the Robertson/Sparck Jones probabilistic retrieval model documents are ranked by this linear function:

$$\sum_{j=1}^{d} x_j \log \frac{p_j(1-q_j)}{(1-p_j)q_j},$$

where p_i and q_i are probabilities to be estimated based on training data (Robertson & Sparck Jones, 1976) or the text of a user request (Croft & Harper, 1979), and the x_j's are binary (1 if a word is present in a document, 0 otherwise).

The classical vector space model (Salton & McGill, 1983, pp. 120–123) (Harman, 1992a), ranks documents using a nonlinear similarity measure called the cosine correlation:

$$\mathrm{SIM}(\mathbf{q}, \mathbf{x}) = \frac{\mathbf{q} \cdot \mathbf{x}}{\|\mathbf{q}\| \, \|\mathbf{x}\|}$$

where $\|\mathbf{x}\| = \sqrt{\sum_{j=1}^{d} {x_j}^2}$, and $\mathbf{q}$ is a query vector with the same features as $\mathbf{x}$. For instance, q_j might be 1 if a word appeared in a textual user request and 0 otherwise, while x_j is the number of times the word occurs in the document of interest, times its inverse document frequency, i.e., a form of *tf* $\times$ *idf* weighting (Salton & McGill, 1983, p. 63), (Harman, 1992a). This model can be recast as linear classification by treating the query as a classifier and incorporating its length normalization into each of the elements of its weight vector:

$$\mathbf{w} = \frac{\mathbf{q}}{\|\mathbf{q}\|}$$

and similarly incorporating the document length normalization into the document vector feature values:

$$\mathbf{x}' = \frac{\mathbf{x}}{\|\mathbf{x}\|}.$$

Indeed, recent work on the vector space model replaces the cosine normalization with other length normalizations, but maintains the linear form of the classifier (Singhal et al., 1996). Many commercial ranked retrieval systems also are based on linear functions, the evaluation of which can be made very efficient via inverted files and other techniques.

Basing an IR system on linear classifiers requires using corpus statistics, the text of a user request, or other knowledge about a class to choose an initial weight vector. These initial values can usually be improved by learning from training data, as discussed in the next section.

3 Algorithms for Training Linear Classifiers

By training a linear classifier, we mean using training data (a set of texts of known class membership) to find a weight vector which accurately classifies new texts. We distinguish between *parametric* and *nonparametric* training algorithms (Duda & Hart, 1973, p. 130). Parametric algorithms use training data to estimate parameters of a probability distribution, and a classifier is produced under the assumption that the estimated distribution is correct. Many probabilistic IR algorithms, for instance the Robertson/Sparck Jones relevance feedback algorithm, are parametric algorithms.

Nonparametric training algorithms do not assume that the training data has a particular distributional form. They instead search directly for a good weight vector, as measured by some *criterion function*. The hope is that the weight vector will generalize well, i.e., that it will also optimize the criterion function, or some other effectiveness measure, on new data.

Different training algorithms can be produced by varying the criterion function and search procedure used (Duda & Hart, 1973, pp. 130-131). Search procedures can operate in either *online* or *batch* fashion. Online algorithms are presented with one training example at a time. They update their current weight vector based on that example and then discard the example, retaining only the new weight vector. Batch algorithms, on the other hand, optimize the criterion function on the entire set of training data at once. Batch algorithms typically do a better job of optimizing the criterion function than online algorithms, and can more easily use criterion functions that are not simple functions of per-example criteria. However, batch algorithms tend to put large demands on memory, and typically require that past training data be saved if additional training is to be done in the future.

In the remainder of this section we review three important nonparametric algorithms for training linear classifiers, showing how they vary in their criterion functions and search procedures. Throughout this section, $\mathbf{x}_i = (x_{i1}, \ldots, x_{id})$ denotes the ith training document, and y_i its associated class label (1 if relevant/a class member, 0 if irrelevant/not a class member).

3.1 The Rocchio Algorithm

The Rocchio algorithm (Rocchio, Jr., 1971; Harman, 1992b) is a batch algorithm. It produces a new weight vector $\mathbf{w}$ from an existing weight vector $\mathbf{w}_1$ and a set of training examples. The jth component w_j of the new weight vector is:

$$w_j = \alpha w_{1,j} + \beta \frac{\sum_{i \in C} x_{i,j}}{n_C} - \gamma \frac{\sum_{i \notin C} x_{i,j}}{n - n_C} \qquad (1)$$

where n is the number of training examples, $C = \{1 \leq i \leq n : y_i = 1\}$ is the set of positive training examples (i.e., members of the class of interest), and n_C is the number of positive training examples. The parameters α, β, and γ control the relative impact of the original weight vector, the positive examples, and the negative examples, respectively. If $\alpha = 0$, $\beta = 1$ and $\gamma = 1$, then $\mathbf{w}/\|\mathbf{w}\|$ is the weight vector of unit length which maximizes

$$\frac{\sum_{i \in C} \mathbf{w} \cdot \mathbf{x}}{n_C} - \frac{\sum_{i \notin C} \mathbf{w} \cdot \mathbf{x}}{n - n_C}, \qquad (2)$$

i.e., the difference in the mean scores for positive and negative training instances. Rocchio refers to such a $\mathbf{w}$ as an optimal query, though he does not show a connection between optimizing the criterion (2) and optimizing more usual effectiveness measures for ranking or binary classification. Since Rocchio was working in a relevance feedback context, he also did not address how well these weight vectors generalize to new data.

Typically, classifiers produced with the Rocchio algorithm are restricted to having nonnegative weights, so that instead of using the raw $\mathbf{w}$ from Equation (1), one uses $\mathbf{w}'$ where

$$w'_j = \begin{cases} w_j & \text{if } w_j > 0 \\ 0 & \text{otherwise.} \end{cases}$$

3.2 The Widrow-Hoff Algorithm

The LMS or Widrow-Hoff algorithm (Widrow & Stearns, 1985, Ch. 6) (Duda & Hart, 1973, p. 156) (here abbreviated WH) is an online algorithm. It runs through the training examples one at a time updating a weight vector at each step. We denote the value of this weight vector before processing the ith training example by $\mathbf{w}_i$. Initially, the weight vector is typically set to the all zeros vector, $\mathbf{w}_1 = (0, \ldots, 0)$; however, other initial settings are possible. At each step, the new weight vector $\mathbf{w}_{i+1}$ is computed from the old weight vector $\mathbf{w}_i$ using training example $\mathbf{x}_i$ with label y_i. The jth component of the new weight vector is found by applying the rule:

$$w_{i+1,j} = w_{i,j} - 2\eta(\mathbf{w}_i \cdot \mathbf{x}_i - y_i)x_{i,j}. \qquad (3)$$

The parameter $\eta > 0$, usually called the *learning rate*, controls how quickly the weight vector $\mathbf{w}$ is allowed to change, and how much influence each new example has on it.

WH is usually viewed as a gradient descent procedure since the term $2(\mathbf{w} \cdot \mathbf{x} - y)\mathbf{x}$ is the gradient (with respect to

$\mathbf{w}$) of the square loss $(\mathbf{w}\cdot\mathbf{x}-y)^2$. Thus, WH tries to move in a direction in which this loss is (locally) decreasing the fastest.

For classifying new instances, it may seem natural to use the final weight vector $\mathbf{w}_{n+1}$. However, there are theoretical arguments (e.g. (Kivinen & Warmuth, 1994)) which suggest that a better choice is the average of the weight vectors computed along the way:

$$\mathbf{w} = \frac{1}{n+1}\sum_{i=1}^{n+1}\mathbf{w}_i. \tag{4}$$

3.3 Kivinen & Warmuth's EG Algorithm

The *exponentiated-gradient* or EG algorithm was introduced by Kivinen and Warmuth (Kivinen & Warmuth, 1994). This algorithm is similar to WH in that it maintains a weight vector $\mathbf{w}_i$ and runs through training examples one at a time. With EG, however, the components of the weight vector $\mathbf{w}_i$ are restricted to be nonnegative and to sum to one. The usual initial weight vector assigns equal weight to all components so that $\mathbf{w}_1 = (1/d,\ldots,1/d)$. The update rule for EG, analogous to Equation (3) for WH, is:

$$w_{i+1,j} = \frac{w_{i,j}\exp(-2\eta(\mathbf{w}_i\cdot\mathbf{x}_i - y_i)x_{i,j})}{\sum_{j=1}^{d} w_{i,j}\exp(-2\eta(\mathbf{w}_i\cdot\mathbf{x}_i - y_i)x_{i,j})}.$$

Thus, each component $w_{i,j}$ is multiplied by $\exp(-2\eta(\mathbf{w}_i\cdot\mathbf{x}_i - y_i)x_{i,j})$, and then the entire weight vector is renormalized. The name of the algorithm comes from the exponentiation of the same gradient that appeared in WH. As before, the learning rate $\eta > 0$ controls the impact of each new training example.

Kivinen and Warmuth give a detailed motivation for both EG and WH. Briefly, the new weight vector $\mathbf{w}_{i+1}$ can be shown to minimize a formula which trades off the conflicting goals of (1) minimizing the loss $(\mathbf{w}_{i+1}\cdot\mathbf{x}_i - y_i)^2$ of the new vector $\mathbf{w}_{i+1}$ on the current example $\mathbf{x}_i$, and (2) penalizing the choice of a new vector $\mathbf{w}_{i+1}$ which is "far" from the old vector $\mathbf{w}_i$. The different rules EG and WH are derived using different choices of distance functions in (2). The parameter η, in their framework, determines the relative importance given to (1) and (2).

3.4 Binary Classification

The algorithms described above produce classifiers which output a numeric value $\mathbf{w}\cdot\mathbf{x}$. This value can be used, for instance, to rank documents or classes for presentation to a user. Something more is needed if binary classification is required, that is, if we must explicitly decide for each document whether it belongs to the class of interest. The approach taken in our experiments is to define a threshold t, and assign a document $\mathbf{x}$ to the class if $\mathbf{w}\cdot\mathbf{x} > t$. The threshold is chosen so as to optimize the desired effectiveness measure on the training set, with the hope that effectiveness on the test set will also be optimized, though this approach has weaknesses (Lewis, 1995a).

4 Error Bounds for WH and EG

Kivinen and Warmuth (Kivinen & Warmuth, 1994) study in detail the theoretical behavior of EG and WH, building on previous work (Cesa-Bianchi et al., 1993; Widrow & Stearns, 1985). Kivinen and Warmuth focus on deriving upper bounds on the error of WH and EG for various settings of the learning rate η. For instance, for the setting of $\eta = 1/(4X^2)$ used in our experiments, and with appropriate assumptions about the random presentation of examples, their results imply the following upper bound on the expected square loss of the vector $\mathbf{w}$ computed by WH:[1]

$$\mathrm{E}\big[(\mathbf{w}\cdot\mathbf{x}-y)^2\big] \le 2\left(\mathrm{E}\big[(\mathbf{u}\cdot\mathbf{x}-y)^2\big] + \frac{\|\mathbf{u}\|^2X^2}{n+1}\right). \tag{5}$$

Here, expectation is with respect to the random presentation of examples $(\mathbf{x}, y)$, X is an assumed upper bound on $\|\mathbf{x}\|$ for all instances $\mathbf{x}$, $\mathbf{u}$ is the vector which gives "best" fit to the data (actually, the bound holds for *all* $\mathbf{u}$), and n as usual is the number of training instances. Thus, the expected square loss of $\mathbf{w}$ is upper bounded by twice the expected square loss of the best vector $\mathbf{u}$, plus a term that is quadratic in the Euclidean length of $\mathbf{u}$ and the maximum length of any instance, but which vanishes at the rate $1/n$.

This bound helps us to predict when WH will perform well (in terms of square loss), namely, when there is some vector $\mathbf{u}$ which fits the data well and when the number of training examples n is large relative to the lengths of $\mathbf{u}$ and of the document representatives.

Kivinen and Warmuth prove bounds of a somewhat different form for EG. With similar assumptions as above and for the setting $\eta = 2/(3R^2)$ used in our experiments, they show that[2]

$$\mathrm{E}\big[(\mathbf{w}\cdot\mathbf{x}-y)^2\big] \le \frac{3}{2}\left(\mathrm{E}\big[(\mathbf{u}\cdot\mathbf{x}-y)^2\big] + \frac{R^2\ln d}{n+1}\right) \tag{6}$$

where $\mathbf{u}$ is the probability vector (nonnegative components summing to one) which best fits the data, and where R is a value such that $\max_j x_{ij} - \min_j x_{ij} \le R$ for all instances $\mathbf{x}_i$.

Note that the "additional term" for EG depends on very different parameters than it does for WH (R and $\ln d$ rather than X and $\|\mathbf{u}\|$). In particular, for a binary representation of documents, this additional term is small even for a huge number of features, since $R = 1$ and the dependence on the number of features d is only logarithmic. It is this mild dependence on the number of features which suggested to us that EG might do well on an IR task.

In sum, Kivinen and Warmuth's results suggest that EG is likely to work well on high dimensional problems. Their results also give insight into how to deal with different document representations. Blum's recent success (Blum, 1995) with a related multiplicative update algorithm on a learning problem with some textual features also encouraged us to try EG.

5 Evaluation Techniques

We tested the algorithms described above on two IR tasks where supervised learning is particularly applicable: categorization and routing. The overall evaluation strategy was similar for the two tasks, and is described in this section. The details of the particular tasks are described in later sections.

Our experiments were of the batch-mode machine learning type. For each data set, a group of classes were defined. A training set of document feature vectors plus class labels

[1] This bound follows from Kivinen and Warmuth's Theorem 5.3 combined with the results in Section 8.

[2] This bound follows from Kivinen and Warmuth's Theorem 5.10 combined with the results in Section 8.

was used by the learning algorithm to produce a classifier for each class. The classifiers were evaluated on a separate test set of document vectors for which class labels were known.

Evaluations of both binary classification and ranking were performed. For binary classification, the weight vector plus a threshold (produced as described in Section 3.4) were used to classify each test document. The effectiveness of this classification was summarized in four contingency table values:

- a = number of class members put in class
- b = number of nonclass members put in class
- c = number of class members not put in class
- d = number of nonclass members not put in class.

Several effectiveness measures can be defined in terms of these values, for instance:

- recall $(R) = a/(a+c)$
- precision $(P) = a/(a+b)$.

We used the F-measure (Lewis & Gale, 1994) (see also (van Rijsbergen, 1979, pp. 173–176)), a weighted combination of recall and precision that can be defined in terms of the contingency table values:

$$F_\beta = \frac{(\beta^2+1)PR}{\beta^2 P + R} = \frac{(\beta^2+1)a}{(\beta^2+1)a + b + \beta^2 c}$$

We use F_β with $\beta = 1$, i.e., $F_1 = 2a/(2a+b+c)$. If a, b, and c are all 0, we define F_1 to be 1.

On the routing data set, we also evaluated the effectiveness of classifiers for ranking. A classifier is applied to each test document, and the documents are sorted by the resulting scores. We measure how close to perfect ranking the classifier came using simple average precision (SAP), which is the mean of precision measured at each class member in the ranking (Harman, 1995b, p. A-9).

6 Text Categorization Task

Text categorization systems classify units of natural language text into pre-defined categories. We describe two new text categorization data sets and how they were used in our experiments.

6.1 The OHSUMED Text Categorization Test Collection

The first collection consists of Medline records from the years 1987 to 1991, distributed as part of the OHSUMED text retrieval test collection (Hersh et al., 1994). For text categorization experiments, we ignore the queries and relevance judgments in the collection, and make use of the MeSH (Lowe & Barnett, 1994) controlled vocabulary terms assigned to the records by National Library of Medicine indexers.

Of the 348,566 OHSUMED records, all but 23 have MeSH categories assigned. These 348,543 records all have titles, but only 233,445 of them have abstracts. Our experiments used only the 233,445 records with both. We used the 183,229 such documents from the years 1987 to 1990 as our training set, and the 50,216 such documents from the year 1991 as our test set.

	Training		Test	
Category	Number	Freq.	Number	Freq.
Set 1				
tickertalk	88	0.0006	35	0.0005
boxoffice	109	0.0008	61	0.0009
nielsens	163	0.0011	85	0.0013
bonds	272	0.0019	115	0.0017
burma	341	0.0024	93	0.0014
ireland	348	0.0024	127	0.0019
quayle	400	0.0028	113	0.0017
dukakis	716	0.0050	20	0.0003
budget	420	0.0029	443	0.0066
hostages	549	0.0038	367	0.0055
Set 2				
yugoslavia	388	0.0027	188	0.0028
aparts	588	0.0041	202	0.0030
dollargold	1053	0.0074	561	0.0084
w.p.w.	1188	0.0083	636	0.0095
german	1231	0.0086	1161	0.0173
gulf	575	0.0040	2896	0.0432
britain/british	2441	0.0171	1074	0.0160
israel	2495	0.0175	1164	0.0174
bush	2553	0.0179	1368	0.0204
japan	2901	0.0203	1436	0.0214

Table 1: TREC-AP categories, separated into Set 1 and Set 2 and sorted by total frequency on the TREC-AP data. We show frequencies on TREC-AP training (years 1988-1989) and test (year 1990) sets. w.p.w. is the category *weather-pageweather.*

MeSH terms consist of a *main heading* optionally flagged with subheadings and importance markers. A total of 14,626 distinct main headings occur in the OHSUMED records. In text categorization research with OHSUMED we have focused on the set of 119 MeSH categories in the Heart Disease subtree of the Cardiovascular Diseases *tree structure* (Lowe & Barnett, 1994). The frequencies of these 119 heart disease categories vary widely, and some in fact do not actually appear in the OHSUMED data. The experiments here used the 49 categories with a training set frequency of 75 or higher, and the 28 categories with a training set frequency between 15 and 74. Results on the remaining 42 categories are omitted here since their high variance requires additional analysis.

The OHSUMED text retrieval test collection was developed by William Hersh and colleagues at Oregon Health Sciences University. It is available by anonymous *ftp* from the server *medir.ohsu.edu* in the directory */pub/ohsumed*. Procedures for the use of OHSUMED in text categorization research were developed by David Lewis and Yiming Yang, with invaluable advice from Christopher Chute, Bill Hersh, Betsy Humphreys, Stephanie Lipow, Henry Lowe, Nels Olson, Peri Schuyler, Mark Tuttle, and John Wilbur. The 119 MeSH Heart Disease categories was extracted by Yiming Yang from the April 1994 (5th Ed.) UMLS CD-ROM, distributed by the National Library of Medicine. Further details are available from Lewis (*lewis@research.att.com*) or Yang (*yiming@cs.cmu.edu*).

6.2 The TREC-AP Text Categorization Test Collection

Our second data set is a subset of the AP newswire stories from the TREC/TIPSTER text retrieval test collection. A total of 242,918 AP stories from the years 1988 through 1990 are included in the collection. In processing this data, we corrected some formatting anomalies in the stories and screened out certain internal editorial notes. We then selected only those stories which had exactly one <HEAD> field (i.e., title) and <TEXT> field (i.e., the body of the article), and meeting other well-formedness criteria. The result was a set of 209,783 AP stories which we call the TREC-AP text categorization test collection.

Several previous text categorization studies with a proprietary AP collection have used two sets of 10 categories: Set 1 (Lewis & Gale, 1994; Cohen, 1995; Cohen & Singer, 1996) and Set 2 (Lewis, 1995b). We have defined these categories on the TREC-AP data set as well (see Table 1). For the experiments reported here, we use the years 1988 and 1989 (142,791 documents) as a training set, and the year 1990 (66,992 documents) as the testing set.

The TREC-AP data covers a different date range than the aforementioned proprietary AP collection, and we use it here with a chronological training/test split rather than a random one. Results on the TREC-AP data therefore cannot be compared to those from the previous AP studies.

The documents in the TREC-AP collection appear on the TIPSTER Information Retrieval Text Research Collection CD-ROMs, Volumes 1 to 3, March 1994 revision. The CD-ROMs are used in the TREC evaluations and are also distributed by the Linguistic Data Consortium (*ldc@unagi.cis.upenn.edu*). Information on TREC is available from Donna Harman (*harman@potomac.ncsl.nist.gov*). Details of the TREC-AP data are available from David Lewis (lewis@research.att.com).

6.3 Details of Experiments

This section summarizes our text categorization experiments, including experimental conditions that were varied.

Feature Extraction. The set of features for each problem was defined by a crude tokenizer that replaced everything but alphabetic characters with a blank, and downcased alphabetic characters. Both binary feature values and cosine-normalized *tf* $\times$ *idf* feature values (SMART *tfc* weights (Salton & Buckley, 1988)) were used for Rocchio and WH, with *idf* estimated on the training set for that run. (This is a deviation from the strict online learning framework.) EG was only run on a binary representation, due to limitations of our current software.

Feature Selection. The full feature set was used in all cases.

Text Segment. The use of titles alone was compared with the use of the main texts (abstract or body of article) alone.

Starting Vector. Rocchio was used without a starting vector ($\alpha = 0$) or, equivalently, a starting vector of $(0, \ldots, 0)$. WH used a starting vector of $(0, \ldots, 0)$and EG a starting vector of $(1/d, \ldots, 1/d)$.

Learning Rate. Rocchio used $\beta = 16$ and $\gamma = 4$, as suggested by Buckley, et al. (Buckley et al., 1994), but with $\alpha = 0$ since no query was used. WH used a learning rate of $\eta = 1/(4X^2)$, where X is the maximum value of $\|\mathbf{x}\|$ in the training set for that run. EG used a rate of $\eta = 2/(3R^2)$, which for a binary document representation is simply 2/3. (See Section 4 for details.)

Training Set. Subsets of 10,000 training documents, as well as the full training set, were tried.

Training Procedure. Training Rocchio is a simple batch process. WH and EG were trained on a single pass through the training set in random order.

Final Classifier. The final Rocchio classifier was used as is. For WH and EG, the mean weight vector across all training examples was used (see Section 3.2 and Equation (4)). In all cases the threshold for binary classification was found by optimizing the F_1 measure on the training set.

Another influence on effectiveness is randomness in the sample used as a training set and, for order sensitive algorithms such as EG and WH, the order in which training instances are presented. We addressed this by running all experiments with ten randomly selected and randomly ordered training sets, and computing average effectiveness over the runs. The same ten randomly ordered sets of training documents were used for all algorithms and categories in a collection.

7 Routing Task

By routing systems, we mean IR systems which provide users access to a stream of texts over a period of time. Examples would be systems that fax newswire stories to a user each morning, which sort incoming email into folders, or which provide a ranked retrieval view of a constantly changing body of text, such as Usenet news. As such, routing systems share characteristics of both retrieval and categorization systems.

7.1 A TREC Routing Data Set

Our routing experiments used data developed in the TREC evaluations (Harman, 1995a). The 741,856 documents from TIPSTER Volumes 1 & 2 were used for training, and the 336,310 documents from Volume 3 were used for testing. (See Section 6.2 for availability.) The TIPSTER distribution includes several sets of "topics" describing the needs of hypothetical users for information. We viewed each such user need as a class to be learned, and conducted routing experiments with this training and test data on two sets of TREC topics: numbers 51-100 and numbers 101-150.

Judgments of which documents belong to each class (i.e., are relevant to each user information need) have been made as part of the TREC evaluations and auxiliary studies, but only a fraction of the documents have been judged. For topics 51-100, a mean of 1,784 training documents (328 relevant and 1,456 nonrelevant) and 2,340 test documents (220 relevant and 2,121 nonrelevant), selected by a pooling strategy (Harman, 1995a), have been judged for relevance. Similarly, for topics 101-150, a mean of 1,252 training documents (233 relevant and 1,019 nonrelevant) and 1,333 test documents (187 relevant and 1,146 nonrelevant) were judged. In our experiments, we train only on the judged training documents.

For the purpose of estimating effectiveness we assume, as do the TREC evaluations, that test documents not judged for a topic are not relevant to that topic. Thus our test set for all topics is of size 336,310.

7.2 Details of Experiments

The routing experiments varied in a number of ways from the text categorization experiments:

Feature Extraction. The set of features was defined by standard INQUERY tokenization of the text, but only words, not phrases were used. The basic INQUERY weighting formula was used (Callan et al., 1995), which has a minimum feature value of 0.4 and weights that tend to be in the range 0.4 to 0.5. Due to this restricted range of values,

Category	Rocc.	WH	EG-bin
Set 1			
tickertalk	.00	.06	.00
boxoffice	.49	.48	.59
nielsens	.52	.51	.46
bonds	.61	.60	.59
burma	.74	.68	.76
ireland	.43	.55	.52
quayle	.79	.78	.77
dukakis	.61	.63	.61
budget	.59	.58	.59
hostages	.58	.60	.58
Set 2			
yugoslavia	.42	.66	.60
aparts	.07	.15	.10
dollargold	.90	.92	.93
w.p.w.	.85	.88	.72
german	.51	.66	.63
gulf	.22	.27	.31
britain/british	.38	.55	.52
israel	.60	.75	.61
bush	.53	.51	.53
japan	.50	.76	.61

Table 2: Per-category effectiveness for Rocchio, WH, and EG on TREC-AP titles. Rocchio and WH use a $tf \times idf$ representation, EG a binary representation. The full training set of 142791 titles is used in all cases. We show mean values (over 10 runs) of F_1.

we trained the WH and EG algorithms with a target output of 0.47 for relevant documents and 0.40 for non-relevant documents. For the EG algorithm this can also be treated simply as a change in the feature values used.

Feature Selection. Time did not allow us to work with the full feature set of words in the routing experiments. (There are 868,795 unique words just in the parsed version of Volumes 1 and 2.) The features used were the content words occurring in the textual description of the topic (on average 7.92 words/topic for topics 51-100 and 8.76 words/topic for topics 101-150), and either 50 or 1000 additional words chosen by a query expansion process similar to that used in the U Mass TREC-4 experiments (Allan et al., 1996).

Text Segment. All textual material was used.

Starting Vector. Rocchio was used with a starting vector of $(0, \dots, 0)$. WH and EG were started with the output of the Rocchio algorithm. EG was also tested with the starting vector $(1/d, ..., 1/d)$, producing similar results (not reported).

Learning Rate. Rocchio was used with parameter settings of $\alpha = 1$, $\beta = 2$ and $\gamma = 0.5$. The ratio of $\beta = 2$ to $\gamma = 0.5$ is the same as in the text categorization experiments. The value $\alpha = 1$ gives some weight to the original topic text, something not available in the text categorization problems. WH was used with a learning rate of $1/\|\mathbf{x}_i\|^2$, that is a different learning rate was used for each example. This difference from the rate used in the categorization experiments is unlikely to have had an effect given the training procedure used (see below). EG used a rate of $\eta = 2/(3R^2)$, with R varying according to the representation used.

Training Set. All documents judged for each topic were used for training.

Training Procedure. Rocchio was trained in the usual batch mode fashion. WH was trained on a sequence of 100,000 examples drawn randomly with replacement from the full training set. EG was trained on 100,000 examples drawn randomly with replacement from either the positive (probability 1/2) or negative (probability 1/2) training examples.

	Topics	
Method	51-100	101-150
INQUERY		
Q+50w / Rocchio	.326	.341
Q+50w / WH	.361	.288
Q+50w / EG	.415	.403
Q+1000w / Rocchio	.203	.190
Q+1000w / WH	.216	.192
Q+1000w / EG	.404	.295
(Buckley et al., 1994)		
Q+50w / Rocchio	.3829	—
Q+500w / Rocchio	.4068	—
(Buckley & Salton, 1995)		
Q+300w/30p : Rocchio	.4045	—
Q+200w/10p : DFO	.4542	—
Q+50w : Rocchio	—	.3471
Q+50w : Best DFO	—	.4078

Table 3: Mean R-precision across routing topics for various training procedures. R-precision is precision at a number of documents equal to the number of relevant documents (Harman, 1995b, p. A-10). w indicates that expansion terms are words, p indicates phrases. DFO is Buckley and Salton's Dynamic Feedback Optimization.

Final Classifier. The final classifier was selected by a pocketing strategy (Gallant, 1986). We pocket (record) the weight vector after 100 training examples. After every 100 subsequent training examples the current weight vector is used to rank the training data and the value of SAP is measured. If the SAP value is higher than that of the pocketed vector, the pocketed vector is replaced by the current vector. At the end of training the current pocketed vector is evaluated on the test data. The threshold for binary classification was found by optimizing F_1 on the training set.

Since the routing experiments always used all training data available, there was no sampling variation. The potential for the ordering of training data to impact effectiveness was slight due to the use of pocketing, and the fact that most examples were examined many times.

8 Results

Table 4 summarizes our results on the three data sets. We compare the overall effectiveness of WH and EG with that of Rocchio in two ways. First, we count the number of classes on which WH (or EG) has a higher F_1 value than Rocchio, and vice versa, as shown in the Wins columns. WH and EG counts are significantly higher ($p < 0.05$) than the corresponding Rocchio counts by a one-tailed sign test (Siegel, 1956, Ch. 5) unless a "?" is shown. Second, we compute the mean F_1 value across classes for each algorithm and compare this in the Mean F_1 columns.

The general pattern of results is as expected. The mean F_1 values hide the usual high variation among classes. The more informative $tf \times idf$ representation is generally superior to the less informative binary one, more training data is better than less, and more positive training instances ("big" categories) is better than fewer. One anomaly is that OHSUMED titles work better than OHSUMED abstracts. The high variance in length of abstracts, with a tendency

Data Set		Num	Num	WH vs. Rocchio		EG vs. Rocchio	
Training	DocRep	Classes	Features	Wins	Mean F_1	Wins	Mean F_1
AP Headline Categorization							
10000	bin	20	40820	$16 > 4$	$(.45 > .33)$	$16 > 4$	$(.44 > .33)$
10000	$tf \times idf$	20	40820	$14 >? 6$	$(.48 > .44)$	$[10 =? 10$	$(.44 > .44)]$
142791	bin	20	40820	$15 > 5$	$(.57 > .40)$	$18 > 2$	$(.55 > .40)$
142791	$tf \times idf$	20	40820	$13 >? 7$	$(.58 > .52)$	$[14 >? 6$	$(.55 > .52)]$
AP Body Categorization							
10000	bin	20	264836	$18 > 2$	$(.48 > .25)$	$18 > 2$	$(.52 > .25)$
10000	$tf \times idf$	20	264836	$15 > 5$	$(.60 > .50)$	$[10 =? 10$	$(.52 > .50)]$
142791	bin	20	264836	$18 > 2$	$(.65 > .33)$	$18 > 2$	$(.61 > .33)$
142791	$tf \times idf$	20	264836	$16 > 4$	$(.72 > .63)$	$[\ 9 <? 11$	$(.61 < .63)]$
OHSUMED Title Categorization (big categories)							
10000	bin	49	64781	$48 > 1$	$(.29 > .15)$	$47 > 2$	$(.29 > .15)$
10000	$tf \times idf$	49	64781	$41 > 7$	$(.34 > .29)$	$[30 >? 19$	$(.29 > .29)]$
183229	bin	49	64781	$47 > 2$	$(.53 > .26)$	$48 > 1$	$(.51 > .26)$
183299	$tf \times idf$	49	64781	$32 > 17$	$(.51 > .47)$	$[35 > 14$	$(.51 > .47)]$
OHSUMED Title Categorization (small categories)							
10000	bin	28	64781	$15 > 4$	$(.04 > .02)$	$23 > 3$	$(.03 > .02)$
10000	$tf \times idf$	28	64781	$21 > 1$	$(.06 > .03)$	$[20 > 7$	$(.03 > .03)]$
183229	bin	28	64781	$26 > 0$	$(.43 > .22)$	$27 > 0$	$(.46 > .42)$
183299	$tf \times idf$	28	64781	$16 > 10$	$(.43 > .41)$	$[21 > 4$	$(.46 > .41)]$
OHSUMED Abstract Categorization (big categories)							
10000	bin	49	135531	$45 > 4$	$(.16 > .07)$	$49 > 0$	$(.27 > .07)$
10000	$tf \times idf$	49	135531	$45 > 4$	$(.28 > .18)$	$[43 > 6$	$(.27 > .18)]$
183229	bin	49	135531	$49 > 0$	$(.51 > .13)$	$48 > 1$	$(.50 > .13)$
183299	$tf \times idf$	49	135531	$44 > 5$	$(.55 > .44)$	$[34 > 15$	$(.50 > .44)]$
OHSUMED Abstract Categorization (small categories)							
10000	bin	28	135531	$13 > 1$	$(.01 > .00)$	$24 > 1$	$(.02 > .00)$
10000	$tf \times idf$	28	135531	$11 > 1$	$(.03 > .00)$	$[24 > 1$	$(.02 > .00)]$
183229	bin	28	135531	$22 > 3$	$(.29 > .10)$	$26 > 0$	$(.39 > .10)$
183299	$tf \times idf$	28	135531	$15 > 12$	$(.39 > .33)$	$[15 > 12$	$(.39 > .33)]$
TREC Document Routing (topics 51-100)							
varies	INQUERY	50	Q+50	$34 > 16$	$(.28 > .22)$	$42 > 8$	$(.36 > .22)$
varies	INQUERY	50	Q+1000	$41 > 9$	$(.16 > .06)$	$49 > 1$	$(.36 > .06)$
TREC Document Routing (topics 101-150)							
varies	INQUERY	50	Q+50	$13 <? 37$	$(.23 < .29)$	$38 > 11$	$(.35 > .29)$
varies	INQUERY	50	Q+1000	$36 > 14$	$(.13 > .07)$	$48 > 2$	$(.19 > .07)$

Table 4: Pairwise comparisons of WH vs. Rocchio, and EG vs. Rocchio. For each condition we show data set, training set size, document representation, number of classes, and number of features. Wins shows the number of classes for which each algorithm had a higher F_1 value. Rocchio is significantly worse by one-tailed sign test unless a "?" is shown. Mean F_1 is the mean value of F_1 across all classes for each algorithm. Results for EG vs. Rocchio on a *tf* $\times$ *idf* representation are bracketed "[]" to indicate that EG was actually run on a binary representation.

toward longer abstracts in later years may be part of the reason.

9 Discussion

Under almost all conditions the Rocchio algorithm was less effective than both WH and EG, sometimes strikingly so. The one notable exception, for which we do not have a good explanation, is on routing topics 101-150, where the WH algorithm does badly for small numbers of features. Similar results (not shown) were obtained when classifiers were used to rank routing documents, with effectiveness measured by SAP. This is despite the fact that WH and EG are online algorithms, and do not optimize a criterion function over an entire training set as Rocchio can.

Document representation had a clear impact on results. Both WH and Rocchio were improved by moving to the more informative $tf \times idf$ representation. Rocchio performed particularly poorly on a binary representation, as has previously been observed (Salton & Buckley, 1990). On the text categorization data EG was run only on a binary representation, as mentioned earlier, but had higher effectiveness than Rocchio running on a $tf \times idf$ representation in many cases. WH and EG also dominated Rocchio on the INQUERY representation used in the routing experiments.

These results are consistent with the theoretical properties outlined in Section 4. For a binary representation, the parameter R which appears in the EG bound (Equation (6)) is equal to 1, and the parameter X^2 in the WH bound (Equation (5)) is equal to the number of (distinct) words in the longest document — at most in the hundreds. For cosine-normalized $tf \times idf$, X is equal to 1. So in each of these cases, the "additional terms" appearing in the WH and EG bounds are quite small given the large number of documents used in our training sets.

Though inferior to EG and WH, it is surprising, in the absence of theoretical guarantees, how well the Rocchio algorithm did with such large feature sets. The per-category data in Table 2 and frequency data in Table 1 suggests Rocchio does its best with relatively low frequency categories.

Our routing experiments show that WH and EG can be used to improve initial weight vectors where both the weights, and terms for which there are nonzero weights, are chosen by the Rocchio algorithm. In addition, EG in particular tends to drive toward zero many of the remaining weights, resulting in a shorter and thus more efficient classifier. This use of EG for term selection is expensive, however. On average, EG with 1000 features was 8 times slower than EG with 50, but gave essentially the same effectiveness. This suggests that when efficiency is a consideration, Rocchio or some other more efficient method be used to choose a limited feature set for which weights are found by EG or WH.

Table 3 compares our results for ranking (rather than binary classifying) the routing data with those of other researchers using the same topics, training data, and test data. For Q+50 features and Rocchio starting weights on topics 101-150, EG does as well as Buckley and Salton's (Buckley & Salton, 1995) computationally intensive Dynamic Feedback Optimization. On topics 51-100, Buckley and Salton's only results used a phrasal representation, and so are not directly comparable, but EG is at least competitive. Buckley, Salton, and Allan (Buckley et al., 1994) found Rocchio better suited to large feature sets on topics 51-100 than we did, probably due to differences in document length normalization.

10 Future Work

There are many improvements possible in our techniques for learning linear classifiers for IR. Applying EG to documents represented by $tf \times idf$ weights on our categorization data is an obvious next step, and other document weighting functions should be investigated as well. The logarithmic dependence of EG on feature set size suggests more radical representation changes. One could combine several variants on stemming, phrase formation, clustering, etc. in the document representation with little danger of overfitting. Cohen and Singer report preliminary results along these lines (Cohen & Singer, 1996).

It is not clear that minimizing squared error on the training set is the best approach to optimizing, for instance, F_β on the test set. The use of general optimization procedures (Buckley & Salton, 1995) is one answer to this problem, but one that sacrifices efficiency and theoretical guarantees. One alternative would be to apply EG to sigmoidal units (Helmbold et al., 1996), which produce probabilities usable for optimization (Lewis, 1995a). Another would be to define error measures for learning which are more tightly coupled with the ultimate effectiveness measure. This may require using a batch mode version of EG, which we in any case wish to compare with other batch mode error minimization procedures (Yang & Chute, 1994).

Maintaining and updating very large weight vectors may take too much space or time, so methods for pruning weight vectors while maintaining theoretical guarantees (Blum, 1995) are also worth examining.

11 Summary

IR methods are being applied to an increasingly broad range of problems, and by implementers who are less experienced with IR systems. Predictability and effectiveness of techniques under a wide range of conditions are important. We have shown that the Widrow-Hoff and EG algorithms for training linear classifiers are not only more effective on IR problems than at least one IR standby, but have a rich theory that lets their performance be better understood and predicted.

Acknowledgments

Thanks to William Cohen, Isabelle Moulinier, Amit Singhal, Yoram Singer, Manfred Warmuth, and Yiming Yang for helpful comments on this work.

References

Allan, J., Ballesteros, L., Callan, J. P., Croft, W. B., & Lu., Z. (1996). Recent experiments with INQUERY. In *Proceedings of TREC-4.*

Blum, A. (1995). Emprical support for Winnow and Weighted-Majority based algorithms: results on a calendar scheduling domain. In *Machine Learning: Proceedings of the Twelfth International Conference*, pp. 124–132.

Buckley, C., & Salton, G. (1995). Optimization of relevance feedback weights. In *SIGIR '95*, pp. 351–357.

Buckley, C., Salton, G., & Allan, J. (1994). The effect of adding relevance information in a relevance feedback environment. In *SIGIR '94*, pp. 292–300.

Callan, J. P., Croft, W. B., & Broglio, J. (1995). TREC and TIPSTER experiments with INQUERY. *Information Processing and Management*, 31(3),327–343.

Cesa-Bianchi, N., Long, P. M., & Warmuth, M. K. (1993). Worst-case quadratic loss bounds for a generalization of the Widrow-Hoff rule. In *Proceedings of COLT-93*, pp. 429–438.

Cohen, W. W. (1995). Text categorization and relational learning. In *Machine Learning: Proceedings of the Twelfth International Conference*, pp. 124–132.

Cohen, W. W., & Singer, Y. (1996). Context-sensitive learning methods for text categorization. In *SIGIR '96*.

Croft, W. B., & Harper, D. J. (1979). Using probabilistic models of document retrieval without relevance feedback. *Journal of Documentation*, 35(4),285–295.

Duda, R. O., & Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. Wiley-Interscience, New York.

Gallant, S. I. (1986). Optimal linear discriminants. In *International Conference on Pattern Recognition*, pp. 849–852.

Harman, D. (1992a). Ranking algorithms. In Frakes, W. B., & Baeza-Yates, R., editors, *Information Retrieval: Data Structures and Algorithms*, pp. 363–392. Prentice Hall, Englewood Cliffs, NJ.

Harman, D. (1992b). Relevance feedback and other query modification techniques. In Frakes, W. B., & Baeza-Yates, R., editors, *Information Retrieval: Data Structures and Algorithms*, pp. 241–263. Prentice Hall, Englewood Cliffs, NJ.

Harman, D. (1995a). Overview of the third Text REtrieval Conference (TREC-3). In (Harman, 1995b).

Harman, D. K., editor (1995b). *Overview of the Third Text REtrieval Conference (TREC-3)*, Gaithersburg, MD 20899-0001. National Institute of Standards and Technology. Special Publication 500-225.

Helmbold, D. P., Kivinen, J., & Warmuth, M. K. (1996). Worst-case loss bounds for single neurons. In *Advances in Neural Information Processing Systems 8*. To appear.

Hersh, W., Buckley, C., Leone, T. J., & Hickman, D. (1994). OHSUMED: an interactive retrieval evaluation and new large test collection for research. In *SIGIR '94*, pp. 192–201.

Kivinen, J., & Warmuth, M. K. (1994). Exponentiated gradient versus gradient descent for linear predictors. Technical Report UCSC-CRL-94-16, Basking Center for Computer Engineering & Information Sciences; University of California, Santa Cruz, CA.

Lewis, D. D. (1995a). Evaluating and optimizing autonomous text classification systems. In *SIGIR '95*, pp. 246–254.

Lewis, D. D. (1995b). A sequential algorithm for training text classifiers: Corrigendum and additional data. *SIGIR Forum*, 29(2),13–19.

Lewis, D. D., & Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *SIGIR '94*, pp. 3–12.

Lowe, H. J., & Barnett, G. O. (1994). Understanding and using the medical subject headings (MeSH) vocabulary to perform literature searches. *Journal of the American Medical Association*, 271(14),1103–1108.

Robertson, S. E., & Sparck Jones, K. (1976). Relevance weighting of search terms. *Journal of the American Society for Information Science*, pp. 129–146.

Rocchio, Jr., J. J. (1971). Relevance feedback in information retrieval. In Salton, G., editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pp. 313–323. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5),513–523.

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

Salton, G., & McGill, M. J. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, New York.

Siegel, S. (1956). *Nonparametric Statistics for the Behavioral Sciences*. McGraw-Hill, New York.

Singhal, A., Buckley, C., & Mitra, M. (1996). Pivoted document length normalization. In *SIGIR '96*.

van Rijsbergen, C. J. (1979). *Information Retrieval*. Butterworths, London, second edition.

Widrow, B., & Stearns, S. D. (1985). *Adaptive Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ.

Yang, Y., & Chute, C. G. (1994). An example-based mapping method for text categorization and retrieval. *ACM Transactions on Information Systems*, 12(3),252–277.