Live and Learn from Mistakes: A Lightweight System for Document Classification

Yevgen Borodin[†] Valentin Polishchuk[#] Jalal Mahmud[‡] I.V. Ramakrishnan[†] Amanda Stent*

[†] Stony Brook University, Comp. Sci.Department, Stony Brook, NY 11790 {borodin, ram}@cs.sunysb.edu
[#] University of Helsinki, Helsinki Institute for I.T., P.O. Box 68 FI-00014 valentin.polishchuk@helsinki.fi
‡ IBM Almaden Research Center, 650 Harry Road, San Jose, CA 95120-6099 jumahmud@us.ibm.com
*AT&T Labs – Research, 180 Park Avenue, Florham Park, NJ 07932 stent@research.att.com

ABSTRACT

We present a Life-Long Learning from Mistakes (3LM) algorithm for document classification, which could be used in various scenarios such as spam filtering, blog classification, web resource categorization, etc. We extend the ideas of online clustering and batch-mode centroid-based classification to online learning with negative feedback. The 3LM is a competitive learning algorithm, which avoids over-smoothing, characteristic of the centroid-based classifiers, by using a different class representative, which we call clusterhead. The clusterheads competing for vector-space dominance are drawn toward misclassified documents, eventually bringing the model to a "balanced state" for a fixed distribution of documents. Subsequently, the clusterheads oscillate between the misclassified documents, heuristically minimizing the rate of misclassifications, an NP-complete problem. Further, the 3LM algorithm prevents over-fitting by "leashing" the clusterheads to their respective centroids. A clusterhead provably converges if its class can be separated by a hyper-plane from all other classes. Lifelong learning with fixed learning rate allows 3LM to adapt to possibly changing distribution of the data and continually learn and unlearn document classes. We report on our experiments, which demonstrate high accuracy of document classification on Reuters21578, OHSUMED, and TREC07p-spam datasets. The 3LM algorithm did not show over-fitting, while consistently outperforming centroid-based, Naïve Bayes, C4.5, AdaBoost, kNN, and SVM whose accuracy had been reported on the same three corpora.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval – *clustering, information filtering.*

General Terms

Algorithms,Performance,Theory,Human Factors,Experimentation.

Keywords

3LM, centroid, clusterhead, classifier, lifelong, online.

1. INTRODUCTION

There is no shortage of machine learning tools for batch-mode data analysis. However, the need to analyze larger and larger datasets created demand for a new class of *online* learning algorithms (online SVMs [7, 29, 33], online decision trees [3], etc.), which can produce similar or better results as their batch-mode counterparts, but are more efficient because they can learn from every new example. In real-life scenarios, training data may not be available all at once (e.g., streaming data), or there may be a prohibitively large training set, which makes batch-mode global optimization algorithms inefficient or even impractical.

Document classification, web resource categorization, spam filtering, blog classification are just some possible applications that could benefit from online learning. Most of well-known document classification algorithms, such as the centroid-based classifier [16, 27], are batch-mode algorithms and cannot be effectively used with streaming data because they need to be rerun periodically on the entire dataset. On the other hand, online algorithms, such as Bayesian Online Classifier [9], require periodic retraining due to "forgetting."

At the same time, a "truly" *online* document classifier should be able to categorize streaming data and adapt to the changing environment by continually learning and unlearning document classes. This type of learning is also called *lifelong* or *never-ending* [4]. For example, spam filtering and classification of news feeds with ever changing features are but a few useful applications of lifelong learning [25, 28].

In this paper, we draw on the existing machine learning techniques for document classification and clustering to extend the batch-mode centroid-based classifier to lifelong learning. We describe the limitations of the centroid-based classifier and propose a new Life-Long Learning from Mistakes (*3LM*) algorithm that:

1. Learns to classify documents on a per-example basis and never stops learning, adapting to evolving data;
2. Uses negative feedback to reinforce learning (learning with a critic) ;
3. Fits the distribution of the data, trying to minimize the number of misclassifications, instead of estimating the centroids (class means);
4. Provably converges for hyperplane-separable classes;
5. Experimentally demonstrates better accuracy than centroid-based document classification algorithms;
6. Demonstrates better performance than centroid-based, Naïve Bayes, C4.5, AdaBoost, kNN, and SVM algorithms;
7. Can find applications in standard document classification tasks, as well as life-long learning scenarios where user feedback may be available, e.g., facebook/twitter feed classification.

The **main contribution** of the *3LM* algorithm is in avoiding over-smoothing[1] present in the centroid-based classifiers. At the same time, the *3LM* algorithm does not overfit for the dataset. It achieves this by using different class representatives – *clusterheads*, which are trained on a per-example basis by an online procedure similar to the perceptron training [23]. We note that the term "clusterhead" has been used earlier in the literature [2, 19] to denote various things not directly related to our use of it.

The rest of the paper is organized as follows. Section 2 overviews related work and provides the background for subsequent

[1] In machine learning, it is common to trade-off between over-smoothing – learning an over-generalized model, and over-fitting – learning a very specialized model that best fits the training data. Techniques such as cross-validation and regularization are often employed to avoid over-fitting.

sections. Section 3 describes our algorithm in detail. Experimental validation of our algorithm appears in Section 4. We discuss impact and significance of our work including possible applications Section 5 and conclude the paper with Section 6.

2. RELATED WORK

2.1 Related Work on Machine Learning

Our work has broad connections with research in online learning, document classification, and reinforcement learning.

Online Learning is a growing subset of machine learning algorithms that learn on a per-example basis. There exist online clustering algorithms [33, 34], online SVM [7, 29, 33], online Bayesian learning [9, 24, 30], etc. The *3LM* algorithm shares some of the features of typical online learning algorithms. Two of its important characteristics are that it learns from negative feedback and never stops learning. In contrast to many online algorithms, it uses a fixed learning rate, but may still converge (see Section 3.3).

Document Classification. A number of methods were proposed for document classification. Examples of document classifiers include: Centroid-based classification [16] (together with the new Class-Feature-Centroid [15]), Naïve Bayesian [26], k-nearest-neighbors (kNN) [10], decision trees (C4.5) [31], neural networks [1, 21], LSI [12], SVM [17, 32], K-means clustering [35], Self-Organizing Maps [22], Boosting [6], Hierarchical [20] to name a few. However, most of these methods are applied to batch-mode classification. Han and Karypis [16] report on a study comparing kNN, Naïve Bayes, and C4.5 decision tree classifiers with the centroid-based classifier, demonstrating that the latter consistently outperforms the other classifiers. In our turn, in Section 4, we present a comparative evaluation of the *3LM* and Centroid-based classifiers, which demonstrates *3LM*'s superiority. We also show that *3LM* outperforms AdaBoost and SVM-based classifiers.

Zhong [7] describes an online document classification algorithm which uses Bayesian approach and hence requires prior knowledge about data distribution. Moreover, the method described in [7] requires an initial training phase. In contrast to [7], *3LM* document classifier does not use any prior knowledge or initial training. Our algorithm is able to learn from negative feedback as it classifies every new example in a stream of training data. Its life-learning ability allows *3LM* to adapt to changing environment by learning and unlearning classes and class assignments over time.

Reinforcement learning [13], also called learning with a critic, involves using some performance feedback to improve the accuracy of a machine learning algorithm. *3LM* also uses negative feedback on the classification accuracy to train the clusterheads. Such feedback is implicit if labeled data samples are used. In other cases, feedback can be provided by human operators. Godbole et al. [14] describe how bringing a human into the loop can significantly improve the accuracy of classification. The *3LM* algorithm can also be used in a scenario involving the users in the classification process.

2.2 Review of Centroid-Based Algorithms

3LM algorithm builds on the ideas of the supervised centroid-based document classification and unsupervised K-means clustering algorithms. In this section we overview their features relevant for the *3LM.*

2.2.1 Batch-Mode Centroid-Based Classification

Centroid-based classifier is one of the simplest algorithms for document classification. In its basic form, the training of the classifier involves computing centroid vectors (class means) by taking the mean of all *tf-idf* adjusted document vectors for each of the classes. That is, for each document class D_k, a centroid is computed as: $\mu_k = \sum_{d \in D_k} d / |D_k|$. Alternatively, instead of being divided by the number of documents, the sum can be normalized. After the classifier has been trained, the centroids are used to classify any new document by computing the cosine similarity between the new document vector d and each of the centroids μ_k:

$$\cos(\mu_k, d) = \frac{\mu_k \cdot d}{\|\mu_k\| \|d\|}$$

The document is then classified as belonging to the class of the highest ranking centroid: $\operatorname{argmax}_{k \in \{1,\dots,K\}}[\cos(\mu_k, d)]$. The computational complexity of training is linear in the number of documents and the terms in them, and the complexity of classification is linear in the number of centroids and the number of terms in the document to be classified. Centroid-based training and classification algorithms have been shown to be both efficient and accurate, with the accuracy consistently outperforming other algorithms such as Naïve Bayes, K-nearest-neighbors, and C4.5, on a range of datasets [16].

2.2.2 Batch-Mode K-means Clustering

K-means is another group of algorithms that use centroids. In contrast to the supervised centroid-based classifier, batch-mode K-means is a popular unsupervised algorithm for clustering document vectors. The algorithm also uses cosine similarity to assign each document to one of the K (initially chosen) centroids. The centroids are then recomputed, and all documents are reassigned to their closest centroid. The algorithm iterates, re-adjusting the positions of centroids until the termination condition is satisfied, which could be: a fixed number of iterations, unchanged cluster assignment, or unchanged centroid positions. K-means clustering is a form of gradient descent, which is known to converge. K-means is popular for its simplicity, speed, and relatively high accuracy. It has been also shown that normalizing vectors to unit length leads to better clustering results [35], while centroids need not have the unit length. The spherical K-means algorithm using vector normalization [11] is one of the fastest document clustering algorithms.

2.2.3 Online K-Means Clustering

One of the optimizations of standard K-means clustering algorithm is learning the centroids *online*, i.e., re-computing them with each new document. This approach is called online K-means clustering and is known to be significantly faster and, at times, more accurate than the traditional batch-mode clustering [34]. Both the batch and online versions of the K-means clustering try to minimize the mean-squared error:

$$E = \frac{1}{N} \sum_d \left\| d - \mu_{k(d)} \right\|^2$$

Where N is the number of documents and $k(d) = \operatorname{argmin}_{k \in \{1,\dots,K\}} \|d - \mu_k\|$ is the index of the centroid $\mu_{k(x)}$ closest to document d. Incidentally, for documents lying on the surface of a hyper-sphere, this is equivalent to maximizing the average cosine similarity between centroids and vectors:

$$C = \frac{1}{N} \sum_d \cos(\mu_{k(d)}, d)$$

where $k(d) = \operatorname*{argmax}_{k \in \{1,\dots,K\}}[\cos(\mu_k, d)]$ is the index of the centroid $\mu_{k(d)}$ most similar to document d. Centroid-based classification uses this same $k(d)$ to determine document class assignment.

2.2.4 *Convergence of Centroid-Based Algorithms*

The batch-mode centroid-based classifier optimizes the same function C as K-means clustering. But since in centroid-based classification every document has a fixed class label $k(d)$, the algorithm exactly computes the absolute maximum of the function. The batch-mode K-means clustering, using the global approach to maximize C, provably converges (possibly to a local maximum) [8]. Online K-means incrementally updates its centroid vectors, also trying to maximize the average cosine similarity C. However, online K-means does not guarantee convergence [34], and, hence, requires that a decreasing learning rate be used while training. We are **not** aware of an online version of the centroid-based classifier. So, we implemented the algorithm to incrementally adjust the centroids in a fashion similar to that of the online K-means clustering $\mu_k += \alpha d_k$, where α is the learning rate. By analogy, convergence of the online centroid-based classifier also depends on a decreasing learning rate.

2.2.5 *Lifelong Learning*

Certain applications such as, for example, classification of streaming documents require that the learning algorithm maintains its flexibility for life. This methodology is called lifelong or never-ending learning. Lifelong learning typically implies a fixed or flexible (as opposed to decreasing) learning rate that will allow the learner to adjust to the changing distribution of the streaming data. In such applications, convergence is not necessarily a desirable property because it may prevent the algorithm from adapting to a changing environment. For example, the online centroid-based classifier with a fixed learning rate will likely not converge even if the streaming documents come from a fixed distribution with separable classes because every new document will shift the position of the centroid. A more serious problem is that the centroid-based classifier requires that every new document be supplied with the correct class label.

3. LIFELONG LEARNING FOR DOCUMENT CLASSIFICATION

In this section, we continue exploring the idea of lifelong learning centroid-based classification and its limitations. We give the intuition for our *3LM* classifier and formally describe its algorithm. We close the section with a proof of convergence of *3LM* for hyper-plane-separable classes.

3.1 Intuition for Learning from Mistakes

3.1.1 *Human-Supervised Classification*

As we have shown, the centroid-based classifier *can* be extended to online learning. However, such an extension would be of little practical use in a lifelong learning scenario because it requires positive feedback for every classified document in order to incrementally maintain the centroids. While it is realistic that humans may supervise document classification and labeling [14], it would be unreasonable to expect them to label every single document. On the other hand, the success of spam filters demonstrates that people are willing to report errors, if it helps improve the accuracy of classification. Thus, an algorithm relying on user feedback should learn from classification mistakes.

3.1.2 *Learning from Mistakes*

Let us now observe the behavior of *3LM*, an algorithm that incrementally learns *only* from misclassified documents. Figure 1 shows a collection of documents projected in 2D, with each document marked as either a black plus or a black minus, based on the class it belongs to. The classes can be separated by a hyper-plane shown by the vertical dashed line. The centroids corresponding to the plus and minus classes are marked as #1.1 red plus and #2.1 red minus, respectively. The centroid-based class assignments are shown with solid ovals, misclassifying two of the plus documents.

If vectors #1.1 and #2.1 are used for classification and are incrementally moved closer to the misclassified documents, then vector #1.1 will eventually migrate to the position marked as #1.2. At that point, vector #1.2 will stop moving (for this fixed distribution of documents) and all documents will be classified correctly, as shown by the dashed circles in Figure 1.

3.1.3 *Centroids vs. Clusterheads*

We call vector #1.2 in Figure 1 a *clusterhead* - a vector representative of a class, which is incrementally updated to be closer to the misclassified documents of its class. The use of clusterheads for document classification instead of centroids is the one of the **key differences** of the *3LM* algorithm from the centroid-based classification. As shown in Figure 1, using centroids for classification causes *over-smoothing* (the opposite of over-fitting), making the class representatives conservatively stick to the center of mass. In contrast, clusterheads allow *3LM* to create a better fit for the distribution of the data, but not over-fit. We will make a clear distinction between centroids and clusterheads for the remainder of this paper.

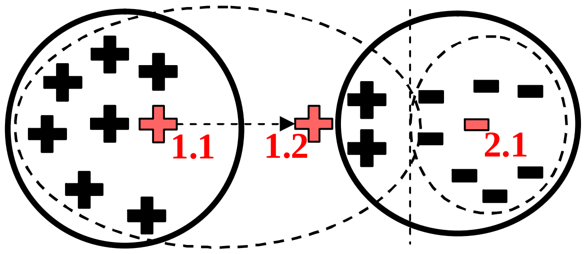

Figure 1 – Centroid (1.1) vs. Clusterhead (1.2)

3.1.4 *Competitive Balancing of Clusterheads*

In Figure 1, the two classes can be separated by a hyper-plane, meaning that the corresponding clusterheads could be positioned to classify all documents correctly. In reality, some classes may create overlaps, where documents of several classes are diffused.

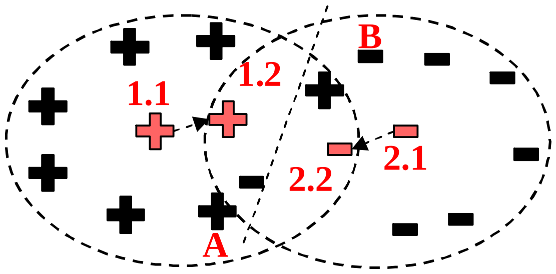

Figure 2 – Balancing in equal-sized classes

Figure 2 shows two overlapped classes of pluses and minuses. We initially place the two clusterheads #1.1 and #2.1 in the centers of mass of their respective classes, and will then incrementally adjust their positions by moving them closer to the misclassified documents. If we randomly (or sequentially) draw and classify documents from the distribution of documents shown in Figure 2, the two clusterheads will then gradually drift towards the misclassified documents to positions #1.2 and #2.2, trying to fit for the error. At that point, getting any closer to the misclassified documents will result in misclassifications of documents A and B, which will, in turn, pull the clusterheads toward the newly misclassified documents. Being equally pulled in the opposite directions, the clusterheads will eventually come to the equilibrium – a "balanced" state where they will "oscillate" between the misclassified documents, competing for dominance.

In Figure 2, the dashed line shows a possible boundary (hyper-plane) minimizing the number of errors. Of course, it is possible that the clusterheads "ovelrap" themselves, each crossing into the other class; in the end of this subsection we discuss how to deal with this (Sections 3.1.9 – 3.1.11).

3.1.5 *Comparing with Centroid-Based Classifier*

It is easy to see from Figure 2 that in the case when the two classes are of comparable density, the centroid-based and clusterhead-based classifiers will have the same accuracy. Indeed, in this case, the dividing line between both the centroids and clusterheads would go through the center of the overlap. However, if class (+) were any denser than class (–), i.e., class (+) had more of its documents in the overlap (Figure 3), then clusterhead #1.2 would push clusterhead #2.2 out of the overlap, resulting in a higher accuracy. In this case, the two correctly classified pluses have more weight than one misclassified minus, which results in clusterhead #1.2 winning over clusterhead #2.2. This competitive behavior exhibited by the clusterheads is an example of *competitive learning* [5, 18].

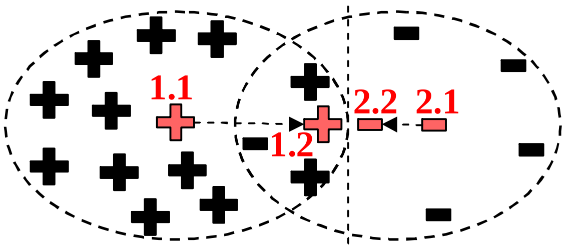

Figure 3 – Balancing in classes of different density

3.1.6 *Squared Error vs. Classification Error*

As discussed in Section 2.1, the centroid-based classifier minimizes the squared error function (maximizes average cosine similarity). However, it is clear (e.g., see Figure 1 and Figure 3) that minimizing the squared error is *not* equivalent to minimizing the number of misclassifications. In contrast to the centroid-based classification, the *3LM* algorithm adjusts the positions of clusterheads trying to fit for misclassified documents and heuristically reduce the actual error rate. Note that finding an optimal solution to the problem of minimizing the error rate is NP-complete (Section 3.3).

3.1.7 *Adapting to Changes*

By continually adjusting positions of the clusterheads, the *3LM* algorithm can adapt to any changes in the distribution of documents. This may prove especially useful in applications, which need to classify documents drawn from an evolving distribution such as news. For example, if in Figure 1 a 3rd class appeared immediately to the left of the class of pluses (not necessarily overlapping), it would result in the clusterhead #1.2 moving left to protect the majority of pluses that would be otherwise misclassified. Equally pulled in all directions, a clusterhead will come to a balanced state.

3.1.8 *Over-Fitting*

Learning from mistakes may cause over-fitting in the traditional machine learning sense when a classifier is first trained until convergence and then used for classification. In a lifelong learning scenario, over-fitting is a transitory state, where a clusterhead tries to fit for a misclassified document, but is immediately corrected by other misclassifications resulting from over-fitting. Our experiments revealed that, in either scenario, our classifier could fit for the training dataset without over-fitting on the testing one. Section 4 describes the experimental results demonstrating *3LM*'s higher recall and precision than that of either batch or online centroid-based classifiers, which use over-smoothed class representatives (centroids).

3.1.9 *Passing of the Clusterheads*

In case of several overlapping classes, each clusterhead may be pulled in opposite directions, oscillating and never approaching its class boundaries. However, in the case of only two overlapping classes, as in Figure 2 and Figure 3, the competitive nature of the clusterheads will make them approach each other. As a result, when positions of competing clusterheads are adjusted to fit for the errors, it may be possible for two clusterheads to pass each other and switch places. Therefore, if left unhandled, it is possible that the two clusterheads may pass each other, resulting in a major misclassification if the documents are classified in batches. For example, users looking through morning news would find that two news categories have the wrong labels.

3.1.10 *Detecting Clusterhead Passing*

If clusterhead #1.2 passes clusterhead #2.2, in either Figure 2 or Figure 3, each clusterhead will become closer to the centroid of the other class. Therefore, if we knew the positions of centroids, we could detect such an event by comparing the distances between the clusterheads and the corresponding centroids. While centroids may not be exactly computed from the misclassified documents, they can be accurately approximated. If we assume that the error rate of classification is relatively low, we can then use the results of the classification and the reported errors to incrementally approximate the centroids.

3.1.11 *Clusterheads on a Leash*

Our solution to clusterhead passing is to pull the contending clusterheads towards their respective centroids. This, in a way, leashes the clusterheads to centroids of their respective classes, allowing the clusterheads to fit, but not over-fit, the errors.

3.2 The *3LM* Algorithm

We now describe the architecture of the *3LM* classifier and present its training algorithm.

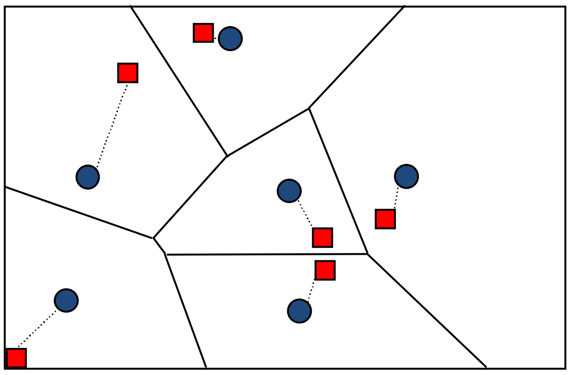

Figure 4 – Voronoi diagram on the clusterheads (red squares). Blue dots are centroid positions.

3.2.1 *The Description of the Classifier*

In the heart of the *3LM* classifier is a standard vector space model with both approximated-centroid and clusterhead vectors continuously learning from training examples and incrementally summarizing document classes. All vectors are normalized to lie on the surface of the unit hyper-sphere to make the computations more efficient and equalize the learning rates. The clusterhead vectors compete for their dominance over the vector space and classify streaming documents based on their proximity. Figure 4 illustrates a Voronoi diagram built on clusterheads shown as red squares. The centroids (blue circles) act as mediators leashing the clusterheads to the center of mass of their respective classes and not letting the competing clusterheads pass each other. The edges of the diagram are the class boundaries: all documents falling into a cell of the diagram are classified according to the class of the cell. Please refer to Figure 1 for an example of why building the Voronoi diagram on centroids, instead of clusterheads, results in higher misclassification rate.

3.2.2 Initialization

The *3LM* algorithm maintains sets $\Omega = \{\omega_1, \dots, \omega_K\}$ and $\mathrm{M} = \{\mu_1, \dots, \mu_K\}$ of normalized clusterheads and normalized (approximate) centroids. Here $1 \dots K$ are the classes of the documents that the classifier has seen so far. Initially, when $K = 0$, both Ω and M are empty. When a document d from a new, previously unseen class arrives, d is added to both Ω and M (and K is increased by 1).

3.2.3 Classification Algorithm

The *3LM* algorithm is similar to the centroid-based classifier in that it also uses vector class representatives to categorize document vectors. However, while the centroid-based algorithm uses centroids to classify documents, *3LM* uses clusterheads to classify them and the approximated centroids to prevent over-fitting:

```
**Algorithm** *classifyDocument*
**Input:** $d$: normalized document vector; $\alpha$: learning rate of centroids;
$\Omega = \{\omega_1, \dots, \omega_K\}$: set of all normalized clusterheads;
$\mathrm{M} = \{\mu_1, \dots, \mu_K\}$: set of all normalized approx. centroids;
$c \leftarrow \underset{k \in \{1,\dots,K\}}{\mathrm{argmax}}[\cos(\omega_k, d)]$
$\mu_c \leftarrow \mu_c + \alpha d$
fixPassing($\{\mu_c\}$, $\Omega$)
**return** $c$
```

The *classifyDocument* algorithm assigns any document d to one of the K classes. The centroid corresponding to the closest clusterhead is incrementally adjusted with a fixed learning rate α: $\mu_c \mathrel{+}= \alpha d$. Later, the same document may be used for adjusting the clusterhead if the document is reported as misclassified.

3.2.4 Training Algorithm

```
**Algorithm** *TrainClassifier*
**Input:** $d$: normalized document vector;
$c$: index of the correct class for document $d$;
$\mathrm{M} = \{\mu_1, \dots, \mu_K\}$: a set of all normalized approximated centroids;
$\Omega = \{\omega_1, \dots, \omega_K\}$: a set of all normalized clusterheads;
$\alpha$: learning rate of centroids;
$\beta$: learning rate of clusterheads;
$e \leftarrow \underset{k \in \{1,\dots,K\}}{\mathrm{argmax}}[\cos(\omega_k, d)]$
**if** $e = c$ **then return**
$\mu_e \leftarrow \mu_e - \alpha d$, fixPassing($\{\mu_e\}$, $\Omega$)
$\mu_c \leftarrow \mu_c + \alpha d$, fixPassing($\{\mu_c\}$, $\Omega$)
$\omega_e \leftarrow \omega_e - \beta d$, fixPassing($\mathrm{M}$, $\{\omega_e\}$)
$\omega_c \leftarrow \omega_c + \beta d$, fixPassing($\mathrm{M}$, $\{\omega_c\}$)
```

For every document d reported as misclassified, we correct the involved centroids and train the clusterheads. Specifically, if c is the correct class of d, but d was (mis-)classified as e, the centroids are corrected with the fixed learning rate α: $\overline{\mu}_c \mathrel{+}= \alpha d$, $\overline{\mu}_e \mathrel{-}= \alpha d$. Reinforcement learning is then used to give positive βd and negative $-\beta d$ rewards to the corresponding ω's : $\omega_c \mathrel{+}= \beta d$, $\omega_e \mathrel{-}= \beta d$. Negative reward is given to accelerate convergence for any clusterhead whose class is hyper-plane-separable from the others. If classes are not separable, the rewards to competing clusterheads will eventually balance out, bringing the clusterheads to a relative equilibrium. To accelerate learning and convergence, a negative reward can be given to all clusterheads found closer to the document than the correct clusterhead: $\omega_{h(d)} \mathrel{-}= \beta d$, where $h(d) = i$: $\cos(\omega_i, d) < \cos(\omega_C, d)$. *TrainClassifier* formally describes the procedure.

3.2.5 Clusterheads on a Leash

The approximated centroids are then used to ensure that the clusterheads do not pass each other, i.e., for every clusterhead ω_i we check if the centroid μ_i of the same class is the closest one to it. If it is not, we adjust the position of ω_i accordingly:

```
**Algorithm** *fixPassing(M,Ω)*
**Input:** $\mathrm{M} = \{\mu_1, \dots, \mu_n\}$: a set of relevant centroids;
$\Omega = \{\omega_1, \dots, \omega_m\}$: a set of relevant clusterheads;
**for** $i \leftarrow 1$ **to** $m$:
    **while** $\cos(\omega_i, \mu_i) \neq \mathrm{argmax}_{j \in \{1,\dots,n\}}[\cos(\omega_i, \mu_j)]$ **do**
        $j \leftarrow \mathrm{argmax}_{j \in \{1,\dots,n\}}[\cos(\omega_i, \mu_j)]$
        // Move $\omega_i$ *towards* $\mu_i$ *so that* $\cos(\omega_i, \mu_i) = \cos(\omega_i, \mu_j)$:
        $\mu \leftarrow \mu_i - \mu_j$
        $x \leftarrow \frac{(\mu_i \cdot \mu)\omega_i - (\mu \cdot \omega_i)\mu}{(\mu_i - \omega_i) \cdot \mu}$
        $\omega_i \leftarrow x / ||x||$
```

The **while** loop of *fixPassing* works as follows: The equation $\cos(\omega_i, \mu_i) = \cos(\omega_i, \mu_j)$ defines the bisector between centroids $\mu_i$and μ_j; the new position of the clusterhead is at the intersection of the segment $\omega_i\mu_i$ with the bisector (that is, the clusterhead $\omega_i$is moved along the segment $\omega_i\mu_i$ until the clusterhead is on the bisector). Figure 5 shows the work of *fixPasing* using Voronoi diagram on the centroids. While there exist pathological cases in which the **while** loop would have to be executed up to n times, in our experiments we never moved the clusterhead more than once.

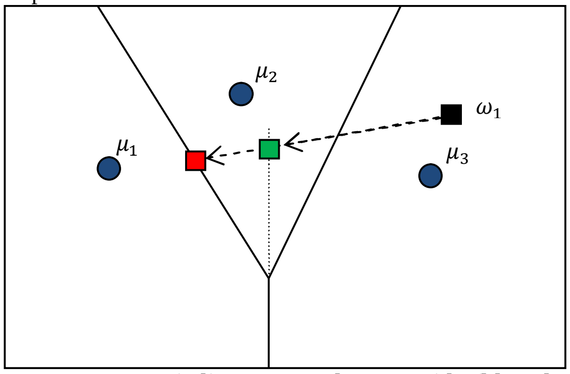

Figure 5 -- Voronoi diagram on the centroids (blue dots). Black square is the initial position of the clusterhead ω_1; it is closer to centroid μ_3 than to centroid μ_1, so *fixPassing* enters the while loop and moves clusterhead ω_1 to the new position (green square) at the bisector between centroids μ_3 and μ_1 (dotted line). Now clusterhead ω_1 is closer to centroid μ_2 than to centroid 2, so *fixPassing* enters the while loop again, and moves clusterhead ω_1 to the bisector between centroids μ_2 and μ_1 (red square). Clusterhead ω_1 is always moved along the line connecting it to centroid μ_1 (dashed); the motion brings the clusterhead to the bisector between the centroid μ_1 and the centroid to which the clusterhead is currently closest.

3.2.6 *Applying tf.idf*

None of the vectors processed by our classifier is normalized using *tf.idf*, which is often used to reduce the weight of words occurring throughout a corpus, thus, reducing the effect of stop words on classification results. As for the document vectors, the distribution of terms across all documents is not available for streaming documents; and incremental computation of *idf* poses additional challenges. We initially removed the 319 most frequent stop words (the exact list can be found in our source code available at http://www.sbhearsay.net/data/3LM.zip) . However, as we show in Section 4.4.3, training and testing the *3LM* classifier with or without stop words has little effect on its overall accuracy.

3.3 Convergence of 3LM

We next prove the convergence of *3LM* for hyper-plane separable classes and give a proof of NP-completeness for the algorithm minimizing the classification error.

3.3.1 *Convergence with Negative Rewards*

Consider the simplest case with only two classes of documents, ***N*** and ***P***, negative and positive. Then, a cosine-similarity classifier consists of two vectors, *N* and *P* – the negative and positive clusterheads. A document *x* is classified as negative (respectively positive) if the angle between *x* and *N* (respectively *P*) is smaller than between *x* and *P* (respectively *N*).

In the most favorable situation, there exists *one* vector *w* such that the angle between *x* and *w* is greater (respectively less) than $\pi/2$ iff *x* is negative (respectively positive). In this case, ***N*** and ***P*** are linearly separable – the (separating) hyper-plane passing through the origin perpendicular to *w*, has the classes on different sides. Without loss of generality, $\|w\| = 1$. Also, without loss of generality, suppose that the documents from ***N*** are negated, i.e., each $x \in \boldsymbol{N}$ is replaced with $-x$. Then, the original classes were linearly separable, if and only if now there exists *w* such that $w \cdot x > 0$ for any *x* in $\boldsymbol{X} \in \boldsymbol{N} \cup \boldsymbol{P}$.

Finding the separating hyper-plane may be viewed as training a perceptron. The classical reinforcement learning scheme for the perceptron is as follows: Starting with $w_0 = 0$, present documents to the perceptron one-by-one; when a document *x* is misclassified, set $w_{i+1} := w_i + x$. This is an example of online learning. It is known that the scheme converges: if $t \equiv \min_{x \in \boldsymbol{X}} w \cdot x$ is the cosine similarity of the "worst" document in ***X***, then after at most $K = 1/t^2$ mistakes we will have w_K aligned with ***w*.**

The negation of documents in ***N*** allows us to consider just one class and look for a hyper-plane passing through origin and having all documents on one side. This motivates us to call this scheme *one-class learning*. Let us now look at how the learning process would work if the documents of class ***N*** were not negated.

The clusterhead ω_i from one-class learning is the clusterhead of the positive class ***P***; denote it by $P_i = w_i$. In one-class learning, the clusterhead of the original negative documents was simply $-w_i$; denote it by $N_i = -w_i$. To see how the learning looks in terms of positive/negative rewards, let us restate the scheme as follows:

If a positive document *x* is misclassified, do $P_{i+1} = P_i + x, N_{i+1} = -P_{i+1} = N_i - x$. If a negative document *-x* is misclassified, do $P_{i+1} = P_i + (-x)$, $N_{i+1} = -P_{i+1} = N_i + x$, that is:

$For\ x \in \boldsymbol{P}: P_{i+1} = P_i + x, N_{i+1} = N_i - x$
$For\ x \in \boldsymbol{N}: N_{i+1} = N_i + x, P_{i+1} = P_i - x$

This can be viewed as a "positive/negative reward rule": On a misclassification, reward the clusterhead of the correct class and punish the clusterhead of the incorrect class.

Let us now consider learning with positive reward only:

$For\ x \in \boldsymbol{P}: P_{i+1} = P_i + x, N_{i+1} = N_i$
$For\ x \in \boldsymbol{N}: N_{i+1} = N_i + x, P_{i+1} = P_i$

Observe that the classes are learned independently: a misclassification of a positive (resp. negative) document has no effect on the negative (resp. positive) clusterhead. In fact, the clusterhead for each class is updated *exactly as in one-class learning*. Thus, the analysis of one-class learning can be carried independently for each class. Moreover, there can be an arbitrary number of classes, with learning of each of them being independent of the others. We summarize this in the following Lemma:

Lemma. *Let* $\boldsymbol{X} = X_1 \cup \ldots \cup X_K$ *be a collection of documents from some* ***K*** *classes. Suppose that for each class k there exists a hyper-plane, passing through the origin, such that all documents from the class lie on the same side of the hyper-plane; i.e., if* w_k *is the unit normal to the plane, for all* $\forall x \in X_k$, $x \cdot w_k > 0$. *Let* $t_k = min_{x \in X_k}\{x \cdot w_k\}$ *be the minimum cosine similarity in class k. Then, after misclassifying at most* $1/t_k^2$ *documents from the class, the lifelong learner will have its clusterhead aligned with* w_k.

With positive rewards only, each class has to present its own number of documents to the classifier. Although the above lemma asserts that it is enough to have just positive rewards in order to ensure convergence, learning can be slower than when negative rewards are also used. This motivated us to use negative rewards in our experiments.

3.3.2 *Minimizing Classification Error is NP-Hard*

Note that even in the simplest case of just 2 document classes, finding *w* that minimizes the number of misclassified documents is an NP-complete problem. Indeed, minimizing the misclassification error is equivalent to finding a vector that satisfies the maximum number of given linear inequalities (in our case, one inequality per each document, and the vector sought is *w* – the normal to the separating hyper-plane), which is an NP-complete problem, called "Open Hemisphere" ([13], p. 246).

4. EMPIRICAL EVALUATION

We conducted a series of experiments to verify the accuracy of *3LM* on the commonly used classification datasets. In our experiments, we compared *3LM* with the other two algorithms discussed in this paper: centroid-based batch and centroid-based online classifiers. In this section, we overview our experiments and show how the obtained results support the intuition and the theory presented in this paper.

4.1 Classification Corpora

Note that the evaluated algorithms were all *single*-topic classifiers, which assign each document to one and only one class. Because of this, and also because the experiments required that all documents be labeled, we ran the experiments on the three datasets widely used for testing single-class document classifiers: 1) Reuters21578 [2] – a collection of news documents; 2) OHSUMED [3] – a collection of medical documents; and 3) TREC07p Spam[4] – a collection of e-mails. From these datasets,

[2] http://kdd.ics.uci.edu/databases/reuters21578/
[3] http://www.nlm.nih.gov/cgi/request.meshdata
[4] http://plg.uwaterloo.ca/~gvcormac/treccorpus07/

we selected a subset of single-topic documents, which we divided into two sets: 80% training and 20% testing.

Our subset of the *Reuters* corpus contained *8,654* news articles covering *65* different topics. There were 65 classes and anywhere from *1* to *3,735* documents per class, so it was possible for some documents to be present in the testing, but not in the training sets, or vice versa. We divided this dataset into five different pairs of *6,924* training and *1,730* testing sets.

Our subset of the *OHSUMED* dataset contained *9,310* medical documents covering *10* different topics. Each of the 10 classes was well represented and contained anywhere from *195* to *3,036* documents. The dataset was divided into five pairs of *7,448* training and *1,862* testing documents.

We chose a fully labeled subset of the *TREC 2007* dataset, which contained *75,419* email messages. However, since we required frequent accuracy sampling for the experiments, we only picked a random subset of *10,000* email messages: *8,031* spam and *1,968* ham emails. These sets were divided into *8,000* training and *2,000* testing documents.

4.2 Experiments on a Fixed Dataset

4.2.1 Experimental Setup

We did 5-fold cross-validation with 80% training and 20% testing documents performing 50 iterations on each of the training datasets. We measured the accuracy on both training and testing datasets after every *1,000* random (w/o replacement) documents used for training. All three datasets was used to train and test the three featured classifiers: centroid-based batch, centroid-based online, and *3LM*. For each classifier-dataset pair, we report recall, precision, and F-measure.

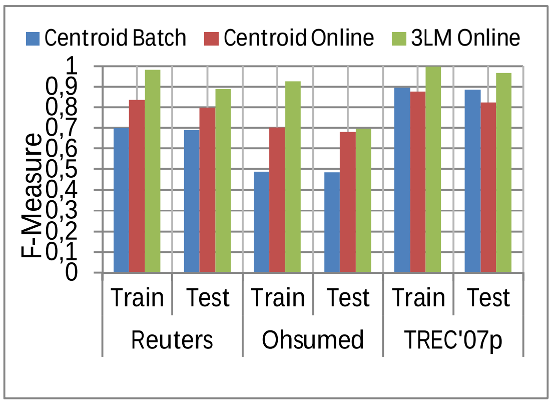

Figure 6 – Performance of Classifiers over Different Datasets

Table 1 – Comparative Evaluation of Classifiers

		Training			*Testing*		
		P	*R*	*F1*	*P*	*R*	*F1*
Reuters	*Batch C*	.800	.621	.699	.797	.611	.691
	Online C	.865	.810	.837	.843	.762	.800
	3LM	**.955**	**.948**	**.983**	**.899**	**.869**	**.890**
Ohsumed	*Batch C*	.049	.485	.487	.486	.483	.485
	Online C	.742	.666	.702	**.723**	.646	.682
	3LM	**.925**	**.925**	**.925**	.697	**.694**	**.696**
TREC07p	*Batch C*	.899	.891	.895	.898	.877	.888
	Online C	.893	.859	.876	.831	.818	.824
	3LM	**.998**	**.998**	**.998**	**.967**	**.964**	**.966**

4.2.2 Summary of Results

The *3LM* algorithm has consistently outperformed both of the centroid-based classifiers on all three datasets. It is noteworthy that our classifier could fit for the training dataset *without* showing any signs of over-fitting on the testing set. Table 1 and Figure 6 summarize the results in terms of the precision (*P*), recall (*R*), and Macro F-measure (*F1*) obtained with the three classifiers on the three datasets used in the experiments. We did not perform significance tests since doing cross-validation removes possible bias of the results.

4.2.3 Batch Centroid-Based Classifier

We first evaluated the batch-mode centroid-based classifier. For the baseline comparison on the Reuters and OHSUMED datasets we would have liked to use the classical paper [16]. However, we decided to re-implement the batch centroid-based classifier for three reasons: (1) Han and Karypis report the overall classification accuracy, while we measure *P*, *R* and *F1*; (2) our methods of dataset preparation are different from those of [16]; (3) performance on TREC07p dataset was not evaluated in [16]. (For the same reasons, we did not validate our implementation of the centroid-based classifier against the implementation of Han and Karypis.) Note, that the classifier over-fits the Ohsumed data showing less than 50% performance on the test part.

4.2.4 Online Centroid-Based Classifier

We then implemented an online version of the centroid-based classifier, as discussed in Section 2.1. Every document vector d_k was used to incrementally adjust the position of the corresponding class-centroid ($\mu_k += \alpha d_k$) using the learning rate of $\alpha = 0.001$. Note that such classifier is not likely to be used in a life-learning scenario because, in order to maintain the centroids, it requires that every document have a label; Conversely, the *3LM* algorithm learns from negative examples, and, hence, needs only the misclassified documents to be labeled. In our experiments, the online centroid-based classifier reached a balanced state after about 400 measurements. It generally achieved higher results than the batch classifier, except for the TREC corpus. The blue and red curves (double lines) in Figure 7, Figure 8, and Figure 9 show the learning curves of the online centroid classifier on the training and testing subsets of the datasets.

4.2.5 3LM Classifier

We then ran our *3LM* classifier on the same datasets, using the same learning rate of $\alpha = 0.001$. After nearly 400 measurements on each of the datasets, the *3LM* algorithm reached the balanced state. The *3LM* algorithm showed no over-fitting on either of the three datasets. The corresponding training (green) and testing (magenta) learning curves can be seen in Figure 7, Figure 8, and Figure 9.

4.2.6 Comparative Analysis

Figure 6 and Table 1 show that the batch-mode centroid-based algorithm performed considerably worse than the online version of the centroid-based classifier on the Reuters and the OHSUMED datasets, but beat the online classifier by *4%* on the TREC07p spam corpus. Figure 7, Figure 8, and Figure 9 show the learning curves of the *3LM* and the centroid-based online classifiers. The learning curves demonstrate how the *3LM* algorithm fits the training data without over-fitting on the testing datasets.

It is notable that the *3LM* algorithm outperformed the centroid-based classifiers in all categories except one (the OHSUMED corpus), on which it showed a *2.6%* lower precision than the online centroid-based algorithm on the testing set of. However, *3LM* showed a *4.8%* higher recall, resulting in *1.4%* higher F-measure than the centroid-based classifier on the same dataset. An interesting observation is that the online-centroid algorithm showed a slight degradation of the overall accuracy after about 100 measurements before reaching a balanced state. Because the centroid-based algorithms tend to produce over-smoothed models, we believe that the degradation was due to the training data.

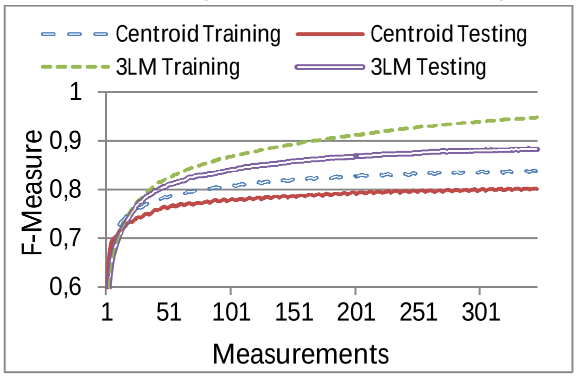

Figure 7-Learning Curve-*3LM* and Centroid Classifiers: Reuters

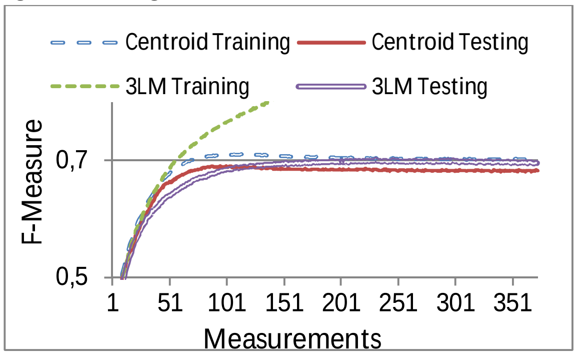

Figure 8 – 3LM and Online Centroid Classifiers: OHSUMED

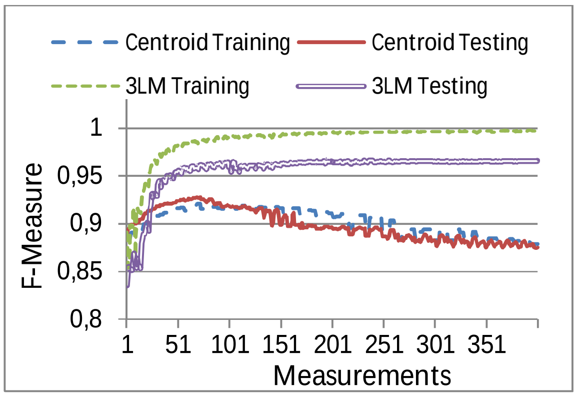

Figure 9 – 3LM and Online Centroid Classifiers: TREC'07p

The online centroid-based classifier showed much worse over-fitting on the TREC07p spam corpus, losing to *3LM* over *10%* in overall accuracy and resulting in the largest spread between the performances of the two algorithms. Both algorithms showed a high degree of volatility of the spam corpus, which we attribute to the nature of the corpus containing a wide variety of documents, including large attachments.

4.2.7 Other Classifiers

We did not directly compare our algorithm to hundreds of other available classifiers, as it would be unpractical. Instead, we used the centroid-based classifier as a benchmark. Han and Karypis [16] reported that a simple batch-mode centroid-based classifier consistently outperformed Naïve Bayes, C4.5, and kNN on a variety of datasets, including Reuters and OHSUMED. Since the *3LM* algorithm achieves much higher accuracy than the centroid-based algorithms on the same datasets, we conclude that *3LM* is likely to outperform Naïve Bayes, C4.5, and kNN. Bloehdorn et al. [6] presented a boosting approach to text classification using higher-level semantic features with AdaBoost on both Reuters and OHSUMED datasets, but achieved lower precision and recall than the *3LM* algorithm. The experiments on the Reuters dataset described in [15] show that SVM-Light, SVMTorch, and LibSVM achieve a Macro-F1 of 0.83, 0.79, and 0.72 respectively, which is substantially less than the 0.89 achieved by the *3LM* algorithm. (It should be noted that we pre-processed the data sets differently from [15]; in particular, we did not give words in titles a ten times higher weight as was done in [15].)

4.3 Life-Long Learning Experiments

4.3.1 Training on the Streaming Data

In all of our previous experiments, the *3LM* algorithm was trained by randomly drawing documents from a *fixed* dataset. To demonstrate how *3LM* can learn from a stream of data, we simulated a stream of news articles that follows a fixed distribution of the Reuters dataset. We "composed" each news article using random words from a randomly chosen class of the Reuters dataset. The news article was then classified and used for training, if misclassified. The overall accuracy was computed for every *10,000* documents.

Note that getting a high accuracy on a streaming *training* set is exactly the intended use of the *3LM* classifier in a lifelong learning scenario. We are, nevertheless, presenting the comparison to the accuracy on the *testing* set *only* to highlight that there is no over-fitting. Figure 10 shows how quickly *3LM* balanced at near-optimal performance on the training set. The lagging but still growing accuracy on the testing dataset can be explained by the fact that some words may not have occurred in the 10 million randomly composed documents. In addition, randomly assembled documents may only be approximating the actual distribution of the news articles in the testing dataset.

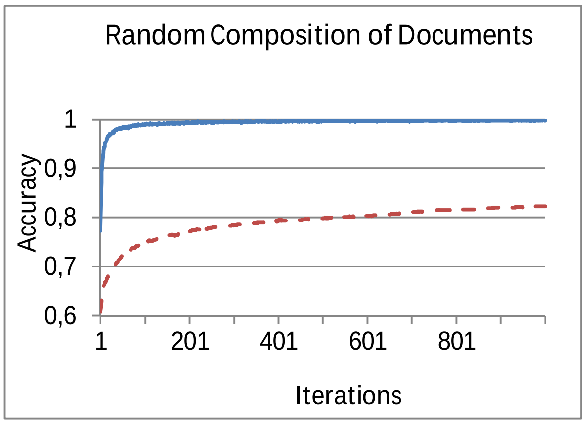

Figure 10 - Simulating streaming news feeds. Solid – training, dashed – testing.

4.3.2 *Evolutionary Experiment*

In the previous experiment, the distribution of documents was fixed. To verify *3LM*'s flexibility, we have designed an experiment that tested the classifier's behavior on a growing dataset with changing distribution. To set up the experiment, we selected 5 largest classes from the Reuters corpus and then trained the classifier on subsets of these 5 classes: we started with one class as the training set and then added the rest of the classes to this training set one by one, at time intervals proportional to the size of the current training set. Periodically, we tested the accuracy of the classifier on the training dataset. Figure 11 shows how *3LM* adapted to the addition of new classes, as the newly-introduced clusterheads entered the competition. The *3LM* algorithm finally converged for these 5 datasets resulting in *100%* accuracy, which means that all classes were hyper-plane separable, as discussed in Section 3.

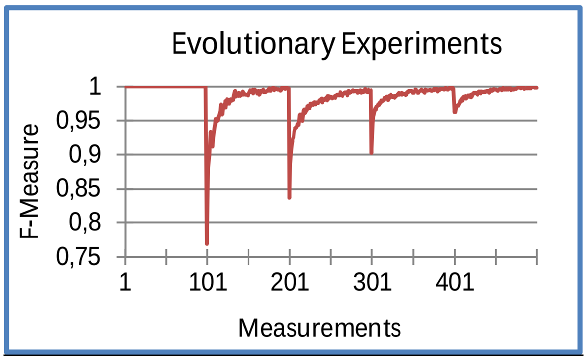

Figure 11 - Training Accuracy with Added Classes

4.4 Robustness of the Algorithm

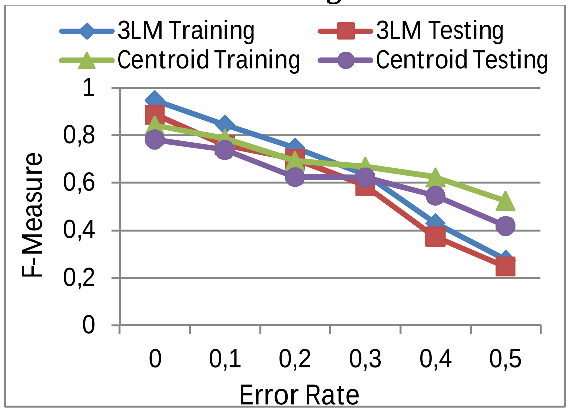

Figure 12 - Error Tolerance for 3LM and Centroid: Reuters

4.4.1 *Tolerance to Random Feedback Errors*

We have also compared how robust both the centroid-based and *3LM* classifiers were in the presence of random errors, which are inevitable in real-life scenarios involving human feedback. The errors were generated by randomly picking a wrong label for every document that was used to train the classifiers. The probability of choosing a certain label depended on the size of the corresponding class; i.e., the label of the larger class had a higher chance of being chosen as the wrong label. The *3LM* algorithm showed a slightly higher rate of degradation of accuracy than the centroid-based classifier. This was not unexpected, since the *3LM* algorithm learns only from the misclassified documents, and hence, receives fewer positive training examples than the centroid-based algorithm. As one can see from Figure 12, at *20%* of the feedback error rate, the *3LM* algorithm was still performing better than the Centroid-based classifier. It is reasonable to expect that human reviewers would do less than *20%* mistakes.

4.4.2 *Variation of the Learning Rate*

In all of our experiments, we used the same learning rate for both positive and negative rewards. We tried varying the positive and negative rewards by a factor of 10 to see the effect of the learning rate on the accuracy of the classifier. As expected, a smaller learning rate (0.0001) slowed down the learning process and, as a result, delayed balancing of the *3LM* classifier, while a larger learning rate (0.01) speeded up learning and balancing, but caused a higher variability. In the end, differing positive and negative rewards had almost no effect on the overall accuracy of the classifier.

4.4.3 *Stop Words*

In our experiments, we did not adjust the term vectors with the *tf.idf*, which could have reduced the effect of stop words on the classification results. Instead, we removed 323 known stop-words. To verify the effect of stop-words on classification accuracy, we ran the *3LM* algorithm on the unfiltered Reuters dataset. As shown in Figure 13, without the stop words, the classification results on the testing set showed only half a percent improvement of the F-Measure, with a minimal effect on the overall accuracy of the *3LM* classifier.

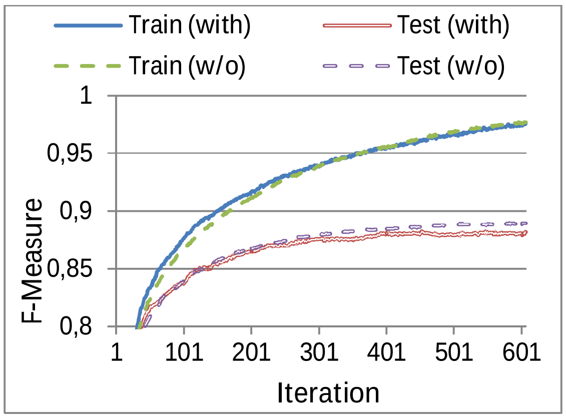

Figure 13 - Effect of Stop Words on 3LM (Reuters)

5. SIGNIFICANCE AND IMPACT

5.1 Applications

Lifelong learning in document classification from streaming data can have a number of applications. For example, spam filters need to keep learning the features of spam messages as spammers are finding new ways to trick the spam filters.

To be able to learn effectively, some classifiers require expert feedback, which is then used to reinforce the learning. This is also called "learning with a critic." Such feedback is implicit if labeled data samples are used; however, in case of classification of streaming data, such as spam, the user involvement is required. In [14], the authors describe how bringing a human into the loop can significantly improve the accuracy of classification. We believe that our algorithm can be viable in classifying streaming data because *3LM* requires only negative feedback, which is usually easier to get, *e.g.*, many users hit the "spam" button to expunge spam from their mailboxes. Here, the users are providing training examples by labeling messages as spam, thus, giving negative feedback and allowing the spam filter to learn from its mistakes.

Another possible application of never-ending learning with *3LM* is sorting the news from a newsfeed, with users providing negative feedback. As we demonstrate in our experiments, news articles can be effectively classified by using only their term vectors. In a fixed dataset scenario, to achieve convergence and finish the training phase, the *3LM* classifier would have to learn with a decreasing learning rate, as is done in online clustering [34]. As we showed in Section 4, *3LM* gives better results than the online and batch-mode centroid-based classifiers. In [16], batch-mode centroid-based classification has been shown to outperform Naïve Bayes, C4.5, and kNN classifiers on the same datasets that we used in our experiments. Furthermore, the experimental results presented in [15] demonstrate that SVM-Light, SVMTorch, and LibSVM do not achieve high accuracy on the Reuters dataset.

5.2 Unique features

To the best of our knowledge, the *3LM* is the first classifier with the ability to adapt to drifts in the document stream while learning from the negative feedback *only* and maintaining low error rate throughout the time. The high performance of the algorithm is due to an enhanced way of representing document classes: for each class not only the centroid of the class' documents is maintained, but also a clusterhead – a more dynamic class representative, better conforming to the class interaction with other classes. The use of clusterheads allows the *3LM* to fit the classes fast; however, overfitting is prevented by leashing the clusterheads to the less agile centroids.

6. CONCLUSION AND FUTURE WORK

In this paper, we described *3LM* – a new algorithm for document classification centered on the idea of lifelong learning from misclassifications. We provided experimental evidence of its effectiveness compared to centroid-based classifiers on the standard Reuters, OHSUMED, and TREC07p datasets.

There are several avenues for expanding our work in the future. From a theoretical point of view, formalizing and analyzing the balancing behavior of the learner is an interesting problem that can shed light on improving the complexity of the algorithm. Properties of our algorithm, including the relationship of balanced state to convergence, evaluation of the likelihood that competing clusterheads may switch places, the effects of **variable** learning rates to achieve faster learning, reducing the oscillations of clusterheads and ensuring better retention of learned weights, deserve further study. The idea of this paper can be expanded to having more than two representatives for a class (in the limit associating each class with the vertices of its convex hull); will this lead to improvement? In addition, features other than inclusion/exclusion of stopwords (e.g., term stems, tf.idf weighted vectors), could be used in the classification.

On the practical side, it will be interesting to explore applications of our algorithm to the emerging paradigm of publish/subscribe systems wherein users express their interests ("subscriptions") to certain kinds of events ("publications"). Such events can conceivably be streams of documents pertaining to different topics of interest. Another intriguing possibility is adapting the idea of clusterheads to K-means clustering. Finally, one may expect better performance from a combined system where *3LM* classifier is used, say in a mixture-of-experts, along with existing models like SVM and NN. Also, a longitudinal user study is required to evaluate *3LM* on real-life streaming data. Last but not least, we believe our ideas can be used also in applications outside the document classification domain.

Acknowledgements. We thank the anonymous reviewers for their many helpful comments which greatly improved the presentation.

7. REFERENCES

[1] Arbib, M.A., *The Handbook of Brain Theory and Neural Networks*. MIT Press, 2002

[2] Basagni, S., K. Herrin, D. Bruschi, and E. Rosti, *Secure pebblenets*, in *Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking \&amp; computing*. 2001, ACM: Long Beach, CA, USA. p. 156-163.

[3] Basak, J., *Online Adaptive Decision Trees: Pattern Classification and Function Approximation*. Neural Comput., 2006. **18**(9): p. 2062-2101.

[4] Beroule, D., *The never-ending learning*. Proceedings of the NATO Advanced Research Workshop on Neural computers, 1988: p. 219-230.

[5] Biehl, M., A. Freking, and G. Reents, *Dynamics of On-line Competitive Learning*. A Letters Journal Exploring the Fronteers of Physics 38(1), 1997: p. 73-78.

[6] Bloehdorn, S. and A. Hotho. *Boosting for text classification with semantic features*. in *Proceedings of the MSW 2004 Workshop at the 10th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2004.

[7] Bordes, A. and L. Bottou, *The Huller: a simple and efficient online SVM*, in *Proceedings of ECML, 16th European Conference on Machine Learning*. 2005.

[8] Bottou, L. and Y. Bengio, *Convergence Properties of the K-Means Algorithms*. Advances in Neural Information Processing Systems, 1995. **7**: p. 585-592.

[9] Chai, K.M.A., H.L. Chieu, and H.T. Ng, *Bayesian online classifiers for text classification and filtering*. Proceedings of SIGIR '02: the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2002: p. 97-104.

[10] Dasarathy, B.V., *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE, 1991.

[11] Dhillon, I.S. and D.S. Modha, *Concept decompositions for large sparse text data using clustering*. Machine Learning, 2001. **42**(1/2): p. 143-175.

[12] Dumais, S., G.W. Furnas, T.K. Landauer, S. Deerwester, and R. Harshman, *Using latent semantic analysis to improve information retrieval*. 1988.

[13] Garey, M.R. and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. 1979, New York, NY: W. H. Freeman.

[14] Godbole, S., A. Harpale, S. Sarawagi, and S. Chakrabarti, *Document classification through interactive supervision of document and term labels*, in *Proceedings of PKDD '04: the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*. 2004, Springer-Verlag New York, Inc.: Pisa, Italy. p. 185-196.

[15] Guan, H., J. Zhou, and M. Guo, *A class-feature-centroid classifier for text categorization*, in *Proceedings of the 18th international conference on World wide web*. 2009, ACM: Madrid, Spain.

[16] Han, E.-H. and G. Karypis, *Centroid-Based Document Classification: Analysis and Experimental Results*. Principles of Data Mining and Knowledge Discovery, 2000: p. 424-431.

[17] Joachims, T., *Text Categorization with Support Vector Machines: Learning with Many Relevant Features*. European Conf. on Machine Learning, 1998.

[18] King, I. and T. Lau. *Competitive learning clustering for information retrieval in image databases*. in *Proceedings of the 1997 International Conference on Neural Networks*. 1997.

[19] Kumar, D., T.C. Aseri, and R.B. Patel, *EECHE: energy-efficient cluster head election protocol for heterogeneous wireless sensor networks*, in *Proceedings of the International Conference on Advances in Computing, Communication and Control*. 2009, ACM: Mumbai, India. p. 75-80.

[20] Li, T., S. Zhu, and M. Ogihara, *Hierarchical document classification using automatically generated hierarchy*. Journal of Intelligent Information Systems, 2007. **29**(2): p. 211-230.

[21] Manevitz, L.M. and M. Yousef, *Document classification on neural networks using only positive examples (poster session)*. SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, 2000: p. 304-306.

[22] Merkl, D., *Document Classification with Self-Organizing Maps*, in *Kohonen Maps*. 1999, Elsevier: Amsterdam. p. 183-197.

[23] Minsky, M.L. and S.A. Papert, *Perceptrons*. 1969, Cambridge, MA: MIT Press.

[24] Opper, M., *A Bayesian approach to on-line learning*. In *On-line learning in neural networks*. Cambridge University Press New York, NY, USA ©1998 ISBN:0-521-65263-4: p. 363-378.

[25] Rennie, J. *ifile: An Application of Machine Learning to E-Mail Filtering*. in *Proceedings of the KDD-2000 Workshop on Text Mining*. 2000.

[26] Rish, I., *An empirical study of the naive Bayes classifier*. 2001.

[27] Rocchio, J., *Relevance Feedback in Information Retrieval*, in *Salton: The SMART Reitrieval System: Experiments in Automatic Document Processing*. 1971, Prentice-Hall. p. 313-323.

[28] Sahami, M., S. Dumais, D. Heckerman, and E. Horvitz, *A Bayesian Approach to Filtering Junk E-Mail*. Learning for Text Categorization: Papers from the 1998 Workshop, 1998.

[29] Sculley, D. and G.M. Wachman, *Relaxed Online SVM for Spam Filtering*. In SIGIR, 2007: *Proceedings of the 30th Annual International ACM SIGIR Conference*.

[30] Solla, S. and O. Winther. *Optimal Perceptron Learning: an Online Bayesian Approach*. In *On-line learning in neural networks*. Cambridge University Press New York, NY, USA ©1998 ISBN:0-521-65263-4.

[31] V.Berikov, A.Litvinenko., *Methods for statistical data analysis with decision tree*. Novosibirsk Sobolev Institute of Mathematics, 2003.

[32] Vapnik, V., *Principles of risk minimization for learning theory*. D. S. Lippman, J. E. Moody, and D. S. Touretzky, editors, Advances in Neural Information Processing Systems 3, 1992: p. 831-838.

[33] Zhang, Z., C. Guo, S. Yu, D.Y. Qi, and S. Long, *Web prediction using online support vector machine*. ICTAI 05, 2005.

[34] Zhong, S., *Efficient online spherical k-means clustering*. Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on, 2005. **5**: p. 3180-3185.

[35] Zhong, S. and J. Ghosh, *Generative model-based document clustering: a comparative study*. Knowl. Inf. Syst., 2005. **8**(3): p. 374-384.