Learning Cumulatively to Become More Knowledgeable

Geli Fei, Shuai Wang and Bing Liu
Department of Computer Science
University of Illinois at Chicago
Chicago, IL, USA
gfei2@uic.edu, shuaiwanghk@gmail.com, liub@cs.uic.edu

ABSTRACT
In classic supervised learning, a learning algorithm takes a fixed training data of several classes to build a classifier. In this paper, we propose to study a new problem, i.e., building a learning system that learns cumulatively. As time goes by, the system sees and learns more and more classes of data and becomes more and more knowledgeable. We believe that this is similar to human learning. We humans learn continuously, retaining the learned knowledge, identifying and learning new things, and updating the existing knowledge with new experiences. Over time, we cumulate more and more knowledge. A learning system should be able to do the same. As algorithmic learning matures, it is time to tackle this *cumulative machine learning* (or simply *cumulative learning*) problem, which is a kind of *lifelong machine learning* problem. It presents two major challenges. First, the system must be able to detect data from unseen classes in the test set. Classic supervised learning, however, assumes all classes in testing are known or seen at the training time. Second, the system needs to be able to selectively update its models whenever a new class of data arrives without re-training the whole system using the entire past and present training data. This paper proposes a novel approach and system to tackle these challenges. Experimental results on two datasets with learning from 2 classes to up to 100 classes show that the proposed approach is highly promising in terms of both classification accuracy and computational efficiency.

Keywords
Cumulative machine learning; lifelong machine learning; unseen classes; open world classification

1. INTRODUCTION
Supervised learning has been very successful in research and in applications. However, existing supervised learning research has focused on developing effective individual statistical algorithms that learn accurate models or classifiers given a fixed dataset. Relatively little research has been done on how to build continuous learning systems that *learn cumulatively* and become more and more knowledgeable as the system sees more and more classes of data over time.

Let us use an example to motivate this research. The 2016 presidential election in the USA has been a hot topic on social media and many social science researchers rely on the collected online user discussions to carry out political science research. During the long campaign, every new proposal made by a presidential candidate is followed by a huge amount of discussions on social media. A multiclass classifier is thus needed to track and to organize the discussions from the general public. As the campaign goes on, the initially built model or classifier will inevitably and frequently encounter new topics (e.g. Donald Trump's plan for immigration reform or Hillary Clinton's new proposal for tax increase) that have not been covered in previous learning. In this case, the classifier must first be able to detect these new topics when they occur rather than classifying them into some existing classes or topics. Second, after enough training examples of new/unseen topics are collected by human users, the existing classifier should incorporate the new classes or topics in the classification system in a manner that does not require rebuilding the whole classification system from scratch.

Based on this example, we can see two inter-related challenges in building a multiclass cumulative supervised learning system: (1) the ability to continuously detect new/unseen classes of data that have not been covered in training by the current classification system, and (2) the ability to cumulatively add new classes to the system without having to re-train the entire system from scratch using all the past training data. In this paper, we aim to solve these two problems in the context of text classification. We call this *cumulative machine learning* or simply *cumulative learning*, which is a special form of *lifelong machine learning* [5, 26, 35, 39]. Formally, cumulative learning is stated as follows:

Problem Statement (**Cumulative Learning**): At any time point *t*, the system maintains a classifier, H^t, learned from a set of past training datasets $D^t = \{D_1, D_2, \ldots, D_t\}$ labeled with corresponding classes (labels) $Y^t = \{l_1, l_2, \ldots, l_t\}$, where every example in each dataset $D_i \in D^t$ is labeled with the same class $l_i \in Y^t$. H^t is able to classify each test instance to either one of the known classes in Y^t or the unknown class l_0, which represents all new or unseen topics. In this case, H^t is said to perform *open world classification*. Once enough training data D_{t+1} has been collected for an unknown topic/class l_{t+1} by the user, H^t is updated to cover the new class l_{t+1} to produce a new classifier H^{t+1} and H^{t+1} is also able to perform *open world classification*. We want to build H^{t+1} upon H^t with minimal efforts and without re-training the entire system.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from Permissions@acm.org.
KDD '16, August 13-17, 2016, San Francisco, CA, USA
© 2016 ACM. ISBN 978-1-4503-4232-2/16/08…$15.00
DOI: http://dx.doi.org/10.1145/2939672.2939835

We now explain why our classification is called *open world classification*. Classic supervised learning assumes that all the test classes have been seen in the training phase and each test instance can only be classified into one of the classes used in training. We say that it makes the *closed world assumption*. In contrast to this closed world assumption, we allow test instances from unknown classes to appear (not seen during training) and our classifier is able to detect such unknown/unseen classes of documents, which is why we call the new paradigm *open world classification*.

Let us now explain why cumulative learning is a form of lifelong machine learning, which is defined as follows [9]:

Definition (**Lifelong Machine Learning (LML)**): At any time *t*, the learner has performed a sequence of *t* learning tasks, T_1, T_2, …, T_t, and has accumulated the knowledge *K* learned in these past tasks. At time *t*+1, it is faced with a new learning task T_{t+1}. The learner is able to make use of the past knowledge *K* to help perform the new learning task T_{t+1}.

Cumulative learning is a form of lifelong machine learning because we can treat task T_{t+1} as the task of learning a multi-class classifier H^{t+1} using all the past and the current data, D_1, D_2, …, D_t, D_{t+1} labeled with corresponding classes, l_1, l_2, …, l_t, l_{t+1}, as well as the past classifier H^t as the knowledge to help train H^{t+1}.

Recently, in the field of computer vision researchers studied the problem of allowing unseen classes of images in the test set, which they call *open set recognition* [18, 31, 32]. We choose to call it *open world classification* (or simply *open classification*) because of the closed world assumption made by classic supervised learning. A framework is also proposed from the risk management perspective [31]. Classic learners define and optimize over the expected empirical risk, which is measured based on errors made on the training data during learning. For open world classification or learning, it is crucial to consider how to extend the model to capture the risk of the unknown by preventing the model from over-generalizing. In order to tackle this problem, [31] introduced the concept of open space risk and formulated an extension of existing one-class and binary SVMs to address the open world classification problem. However, their proposed method is weak as the positively labeled open space is still an infinite area. We have addressed this problem in [14], which we adopt for this work, and extend it further to cumulative learning.

Following the formulation of open world learning, we discuss a solution for detecting new classes of documents by reducing the open space risk while balancing the empirical risk in learning. Intuitively, given a target class of documents, the positive open space for the class is considered to be the space that is sufficiently far from the center of the target class documents. In the multiclass classification setting, each of the classes can be regarded as a target class and should be surrounded by a ball covering the target class area, while any document falling outside of the balls of all target classes is considered belonging to unseen/new classes. To build an open world classifier, a recent learning technique called *center-based similarity space learning* (CBS learning), which was originally proposed by Fei and Liu [13] for solving a negative covariate shift problem, is employed to give an initial solution to the proposed open space formulation, which significantly reduces the open space risk compared to that of [31].

Being able to detect unseen classes/topics is still insufficient for a multiclass classifier to handle the growing number of topics of interest. We need to incorporate the detected new classes into the system with minimal effort. A naive approach to solving this problem is to re-train the entire system including the new class of data from scratch. However, such a solution is only feasible if the number of classes is small. It is inappropriate when the number of classes grows very large.

In this work, we propose a new learning strategy, which is inspired by the process of human concept learning, to make an attempt to tackle the *cumulative learning* problem. Human beings are exposed to new concepts all the time. One particular way we learn a new concept is by searching from the already known concepts, looking for similar ones, and then trying to find the difference between these known concepts and the new one without using all the known concepts. For example, assume we have already learned concepts like "movie", "furniture," and "soccer". Now we are presented with the concept of "basketball" and its documents. We find that "basketball" is similar to "soccer", but very different from "movie" and "furniture". Thus we just need to *accommodate* the new concept "basketball" into our old knowledge base by focusing on distinguishing the "basketball" and "soccer" concepts. We do not need to worry about the difference between "basketball" and "movie" or "furniture", because the existing perception or concepts of "movie" and "furniture" can easily tell that documents from "basketball" do not belong to either of them. Based on the above possible human learning process, the proposed learning process adds a new class of documents to the system that only disturbs a small subset of the past classes.

As we indicated above, cumulative learning is related to lifelong learning [5, 26, 35, 39] because we aim to perform learning continuously to make the system more and more knowledgeable, which is analogous to human learning. However, it is different from current lifelong machine learning [35] and transfer learning [25] methods because existing work in these areas mainly focuses on knowledge transfer, i.e., how to make use of the past data or knowledge to help new learning tasks. None of the methods is able to detect unseen classes or incrementally update an existing classifier, which make cumulative learning require a different type of algorithms. Our problem is also different from existing research in online learning and incremental learning [1, 6, 42]. Online learning aims to handle new instances of the known classes, while we focus on handling unseen/new classes of documents by recognizing them and updating the existing classification system.

In summary, this paper makes the following contributions:

1. It proposes the new learning problem of *cumulative learning*, which presents a new form of lifelong learning. It involves two unique challenges, detecting and learning new knowledge (classes or concepts) over time so that the system becomes more and more knowledgeable. The first challenge is called open world classification and the CBS learning framework is adopted to solve the problem in a similarity space.
2. It proposes a learning strategy for cumulatively adding new classes of documents into the existing classification system without requiring re-training the whole system from scratch.
3. Extensive experiments show that the proposed method gives superior results compared to state-of-the-art baselines in terms of both classification accuracy and learning efficiency.

2. Related Work

Our work addresses an issue that is related to and has received attention from various machine learning paradigms such as open set recognition, lifelong learning, transfer learning, multi-task learning, and online learning. We compare with them below.

2.1 Open World Classification

Compared to research on multiclass classification with closed world assumptions, there is relatively less work on open world classification. In this subsection, we review related work on one-class classification, SVM decision score calibration, open set recognition, and others.

One-class classifiers, which only rely on positive training data, are natural starting solutions to the open world multiclass classification task. One-class SVM [33] and SVDD [37] are two representative one-class classifiers. One-class SVM treats the origin in the feature space as the only member of the negative class, and maximizes the margin with respect to it. SVDD tries to place a hypersphere with the minimum radius around almost all the positive training points. [22] demonstrated that SVDD and one-class SVM are comparable when the Gaussian kernel is applied. However, as no negative training data is used, one-class classifiers have trouble producing good separations, leading to poor results.

This work is also related to using thresholded probabilities for rejection. As the decision score produced by SVM is not a probability distribution, several techniques have been proposed to convert a raw decision score to calibrated probabilistic outputs [2, 12, 17, 27, 41]. Usually a parametric distribution is assumed for the underlying distribution, and raw scores are mapped based on the learned model. A variation of the approach in [27] is the most widely used probability estimator for SVM score calibration. It fits a sigmoid function to the SVM scores during training. Provided with a threshold, a test instance can be rejected if its highest probability of belonging to any class is lower than the threshold.

Recently, researchers in computer vision [18, 31, 32] made attempts to solve open world classification and proposed the concept of open set recognition for visual learning. [31] introduced the concept of open space risk, and defined it as a relative measure. The proposed model reduced the open space risk by replacing the positively labeled half-space of a binary linear classifier with a positive region bounded by two parallel hyperplanes. While the positively labeled region for a target class is reduced compared to the half-space in the traditional linear SVM, its open space risk is still infinite. In [18], the authors proposed to use Extreme Value Theory (EVT) to estimate the unnormalized posterior probability of inclusion for each class by fitting a Weibull distribution over the positive class scores from a pre-trained 1-vs-rest multiclass RBF SVM classifier. [32] introduced the Compact Abating Probability (CAP) model, which explained how thresholding the probabilistic output of RBF One-class SVM manages the open space risk. Using the probability output from RBF one-class SVM as a conditioner, the authors combined RBF One-class SVM with a Weibull-calibrated SVM similar to the one in [18]. For both methods [18, 32], decision thresholds need to be chosen based on the prior knowledge of the ratio of unseen classes in testing, which is a weakness of the methods.

[11] proposed Exploratory Learning in the multiclass semi-supervised learning (SSL) setting. In this work, an "exploratory" version of expectation-maximization (EM) is proposed to extend traditional multiclass SSL methods, which deals with the scenario when the algorithm is given seeds from only some of the classes in the data. It automatically explores different numbers of new classes in the EM iterations. The underlying assumption is that a new class should be introduced to hold an instance x when the probability of x belonging to the existing classes is close to uniform. This is quite different from our work. First, it works in the semi-supervised setting and assumes that test data is available during training. Second, it only focuses on improving accuracy on the classes with seed examples. Our work in [14] dealt with the problem using an entirely different approach adopted from [13]. However, these works did not propose or deal with cumulative learning, which is important for an intelligent system as it allows the system to learn more and more and become more and more knowledgeable.

2.2 Lifelong Machine Learning

Our work is related to lifelong machine learning [5, 26, 35, 39]. In the context of supervised learning, early work on lifelong learning focused on transferring invariances in neural networks. For example, memory-based and explanation-based neural networks (EBNN) based methods were proposed in [38, 39], which transferred knowledge across multiple learning tasks. Their learning task was similar to ours, but they mainly focused on helping the classification of the new class. Also, their methods were also inefficient. [36] made some improvements in terms of efficiency to those in [38, 39], but the framework was similar. [30] proposed the Efficient Lifelong Learning Algorithm (ELLA). [29] further enhanced ELLA through actively selection of the next task to learn. However, each of ELLA's learning task is independent of others, i.e., each task's learning and testing are not related to others. Thus, it solves a different problem. This is also the case for the lifelong supervised learning in [9]. Clearly, none of previous works detects new or unseen classes. Our work is complementary to existing research. Lifelong learning has also been conducted in reinforcement learning [5], and unsupervised topic modeling [8, 9], which use the knowledge extracted from past documents of many domains to improve topic discovery in future tasks.

2.3 Online and Incremental Learning

Online learning and incremental learning [1, 6, 10, 15, 20, 42] mainly aim at handling new instances of known classes. In both scenarios, new data instances belonging to the known classes and their class labels are incrementally revealed. The goal of online learning is to make a sequence of accurate predictions in an online manner given the knowledge of the correct answers to previous prediction tasks. However, our problem has a different setting, in which a new class of documents arrives together and online updating is not required. We also detect new classes and update the learned classifier without re-training the entire system. Although [40] allows new classes of data to be incrementally added, the paper does not detect new/unseen classes, which makes the system less applicable in real-world applications.

3. Cumulative Learning

This section presents the proposed learning strategy and process to solve the cumulative learning problem. As discussed in the introduction section, the proposed learning process is similar to that of human concept learning. It cumulates knowledge and uses the cumulated knowledge to help update the existing classification model and to accommodate the new class, so that the new classification model can classify both existing classes and the new class, as well as detecting unseen classes constantly. The proposed method is based on the 1-vs-rest strategy of SVM. This section focuses on the overall algorithm and how to incorporate the new class with minimum effort in training by exploiting the existing classification model and the past data as the prior knowledge. The underlying learning method will be discussed in more details in Section 4.

3.1 Training a Cumulative Classification Model

We already have an open world classification system at time t with its classification model $H^t = \{h_1, h_2, \dots, h_t\}$ built for the past t classes $Y^t = \{l_1, l_2, \dots, l_t\}$ using their corresponding training sets $D^t = \{D_1, D_2, \dots, D_t\}$. At time $t+1$, the new dataset D_{t+1} of class l_{t+1} arrives, and the classification model H^t needs to be updated or extended to produce a new classification model H^{t+1}. We note that each h_i in H^t or H^{t+1} is a 1-vs-rest SVM classifier built using the CBS learning method in [13] for class l_i treating l_i as the positive class. We will discuss CBS learning in the next section.

Specifically, the learning system goes through the following two steps to update the current state of the classification system H^t to build a new one H^{t+1} that can classify test data from all classes in $\{l_1, l_2, \dots, l_t, l_{t+1}\}$ as well as detect unseen classes of documents denoted by l_0.

1. Searching for a set of similar classes SC that are similar to the new class l_{t+1}.
2. Learning to separate the new class l_{t+1} from the classes in SC.

In order to perform the first step, we need a way to measure the similarity between classes. There are many possible ways. One way is to perform clustering every time when a new class arrives and see which cluster the new class falls in. However, it is difficult to set the number of clusters as the number of classes changes over time. It is also hard to know how the classes in the same cluster are related in the overall classification problem. Another way to measure the similarity between classes is by computing the similarity between the centers of each class of documents. However, this approach does not know the spread of each class of documents and again, it is not clear how this distance is related to the final classification.

In this work, we propose to quantify the similarity between a new class l_{t+1} and the existing ones $l_1, l_2, \dots, l_t$ by running each of the 1-vs-rest binary classifiers in $H^t = \{h_1, h_2, \dots, h_t\}$ to classify instances in D_{t+1}. Those past classifiers that accept (classify as positive) a certain number/percentage λ_{sim} of instances from D_{t+1} are regarded as similar classes and are denoted by SC. This method is intuitive because if a past classifier h_i classifies many instances in D_{t+1} as positive, it means that the two classes of data are close to each other and need to be separated subsequently.

Separating the new class l_{t+1} and classes in SC actually involves two steps:

1. Building a binary classifier h_{t+1} for the new class l_{t+1}. It is intuitive to build h_{t+1} for class l_{t+1}, using D_{t+1} as the positive training data and the data of the classes in SC as the negative training data.
2. Updating the existing classifiers for the classes in SC. The reason for the updating is that the addition of class l_{t+1} confuses those classifiers in SC. To re-build each existing classifier h_i, the system needs to use the original negative data employed to build the existing classifier h_i and the new data D_{t+1} as the new negative training data. We still need the old negative training data because we want the new classifier still to be able to separate class l_i from those old classes.

The detailed algorithm is given in Algorithm 1. Line 1 initializes SC to the empty set. Line 3 initializes the variable CT (for count) to record the number of instances in D_{t+1} that are classified as positive by classifier h_i. Lines 4-9 use h_i to classify each instance in D_{t+1} and record the number of instances that are classified (or accepted) as positive by h_i. Lines 10-12 check whether there are too many instances in D_{t+1} that have been classified as positive by h_i to render class l_i as similar to class l_{t+1}. λ_{sim} is a threshold controlling how many percent of instances in D_{t+1} should be classified to class l_i before considering l_{t+1} as similar/close to class l_i. Lines 14-17 build a new classifier h_{t+1} and update all the classifiers for classes in SC.

Algorithm 1. **Cumulative Learning**

Input: Classification model $H^t = \{h_1, h_2, \dots, h_t\}$ till time t
Past dataset $\{D_1, D_2, \dots, D_t\}$ till time t
New dataset D_{t+1} at time $t+1$
Similarity threshold λ_{sim}
Output: Updated classification model $H^{t+1} = \{h_1, \dots, h_t, h_{t+1}\}$

```
1:  SC = empty;
2:  for each classifier h_i ∈ H^t do
3:    CT = 0;
4:    for each test instance d_j ∈ D_{t+1} do
5:      class = h_i(d_j) // classify doc d_j using h_i
6:      if class = l_i then // wrongly classified
7:        CT = CT + 1
8:      end-if
9:    end-for
10:   if CT > λ_sim * |D_{t+1}| then
11:     SC = SC ∪ {l_i}
12:   end-if
13: end-for
14: build h_{t+1} and add to H^{t+1}
15: for each h_i of class l_i ∈ SC do
16:   update h_i
17: end-for
18: return H^{t+1}
```

In summary, the proposed learning process uses the set SC of similar classes to the new class l_{t+1} to control both the number of classifiers need to be built/updated at time $t+1$ and also the number of negative instances used in building the new classifier h_{t+1}. It thus greatly reduces the time compared to that of training a 1-vs-rest multiclass classifier using all the data. However, running existing classifiers to classify instances from the new class will cause some overhead. But the overhead is small compared to the training time needed when the number of classes is very large.

Combining the cumulative learning process proposed in this section and the *cbsSVM* learning method discussed in Section 4 as the underlying learner, our system (which is able to handle both challenges in cumulative learning) is called *CL-cbsSVM* (*CL* stands for *Cumulative Learning*).

3.2 Testing the Cumulative Classification Model

To test the new classification model $H^{t+1} = \{h_1, h_2, \dots, h_t, h_{t+1}\}$, we follow the standard technique of combining the set of 1-vs-rest binary CBS classifiers in H^{t+1} to perform multiclass classification with a rejection option for the unknown. As output scores from different SVM classifiers are not comparable, the SVM scores for each classifier are first converted to probabilities based on a variant of Platt's [27] algorithm, which is supported in LIBSVM [7]. Let $P(y|\mathbf{x})$ be a probabilistic estimator, where $y \in Y^{t+1} (= \{l_1, l_2, \dots, l_t, l_{t+1}\})$ is a class label and $\mathbf{x}$ is the feature vector of a test document. Let θ be the decision threshold (we use 0.5) and y^* be the final predicted class for $\mathbf{x}$. Recall we use l_0 to represent all possible unknown classes. The classification on the test document $\mathbf{x}$ is done as follow:

$$y^* = \begin{cases} argmax_{y \in Y^{t+1}} P(y|\mathbf{x}) & if\ P(y^*|\mathbf{x}) \geq \theta \\ l_0 & otherwise \end{cases}$$

The idea is that for the test instance $\mathbf{x}$, each binary classifier $h_i \in H^{t+1}$ is used to produce a probability $P(l_i|\mathbf{x})$. If none of the probabilities is greater than θ (= 0.5), the document represented by $\mathbf{x}$ is regarded as unseen or belonging to l_0; otherwise it is classified to the class with the highest probability.

4. Open Learning for Unseen Class Detection

This section gives an overview of the CBS learning method given in [13]. It performs binary classification focusing on identifying positive class documents and has a superior capability of detecting unseen classes or classifying them as not positive. It provides the base classification method for our cumulative learning. Below, we first discuss the open space risk management strategy in [14], and then apply an SVM-based CBS learning method [13] as a solution towards the open space risk management strategy. Although CBS learning only performs binary classification with the positive class as the class of interest, applying the 1-vs-rest method described in Section 3.2 gives us a multiclass CBS classification model, which is called *cbsSVM* in [14].

4.1 Open space risk formulation

Consider the risk formulation for open world classification by Scheirer et al. [31], where apart from empirical risk, there is risk in labeling the open space (space away from positive training examples) as "positive" for any unknown class. Due to the lack of information of a classification function on the open space, open space risk is approximated by a relative Lebesgue measure [34]. Let S_o be a large ball of radius r_o that contains both the positively labeled open space O and all of the positive training examples; and let h_y be a measurable classification function where $h_y(\mathbf{x}) = 1$ for recognition of class y of interest and $h_y(\mathbf{x}) = 0$ otherwise. In our case, y is simply any class of interest l_i.

In [14], O is defined as the positively labeled area that is sufficiently far from the center of the positive training examples. Let $B_{r_y}(cen_y)$ be a closed ball of radius r_y centered around the center of positive class y (cen_y), which, ideally, tightly covers all positive examples of class y only; S_o be a larger ball $B_{r_O}(cen_y)$ of radius r_o with the same center cen_y. Let classification function $h_y(\mathbf{x}) = 1$ when $\mathbf{x} \in B_{r_O}(cen_y)$, and $h_y(\mathbf{x}) = 0$ otherwise. Also let q be the positive half of the space defined by a binary decision hyperplane Ω obtained from a SVM classifier trained using positive and negative training examples. We also define the size of ball B_{r_O} be bounded by Ω, i.e., $B_{r_O} \cap q = B_{r_O}$. Open space is defined as,

$$O = S_o - B_{r_y}(cen_y)$$

where radius r_O of S_o needs to be determined during learning for each known positive class.

This open space formulation greatly reduces the open space risk compared to those of traditional SVM and 1-vs-Set Machine in [31]. Traditional SVM uses classification function $h_y^{SVM}(\mathbf{x}) = 1$ when $\mathbf{x} \in q$, and its positive open space being approximately $q - B_{r_y}(cen_y)$, which is only bounded by the SVM decision hyperplane Ω. For 1-vs-Set Machine in [31], it has classification function $h_y^{1-vs-set}(\mathbf{x}) = 1$ when $\mathbf{x} \in g$, where g is a slab area with thickness δ bounded by two parallel hyperplanes Ω and Ψ ($\Psi \parallel \Omega$) in q, and its positive open space is approximately $g - B_{r_y}(cen_y)$. Given open space formulations of the traditional SVM and 1-vs-Set Machine, we can see that both methods label an unlimited area as positively labeled space, while the formulation in [14] reduces it to a bounded spherical area.

Given the above open space definition, the question is how to estimate S_o (or the radius r_O) for the positive class. The center-based similarity space (CBS) learning proposed by Fei and Liu [13] is suitable for the purpose, which was originally proposed to deal with negative covariate shift. Below, we introduce CBS learning and briefly discuss why it is suitable for the new problem.

4.2 Center-Based Similarity Space Learning

Let $D = \{(d_1, y_1), (d_2, y_2), \ldots, (d_n, y_n)\}$ be the set of training examples, where d_k is a document and $y_k \in \{1, -1\}$ is its class label. In traditional classification, each d_k is represented with a feature vector $\mathbf{x}_k$, which we call a *document space vector* (*ds-vector*), and D is directly used to build a binary classifier. However, CBS learning transforms each *ds*-vector $\mathbf{x}_k$ (no change to its class label) to a center-based similarity space feature vector (CBS vector) $\mathbf{cbs\text{-}v}_k$. Each feature in $\mathbf{cbs\text{-}v}_k$ is a similarity between the center $\mathbf{c}_j$ of the positive class documents and $\mathbf{x}_k$. A classifier can be build based on CBS vector representations of documents in D, i.e., each $\mathbf{x}_k$ is replaced with $\mathbf{cbs\text{-}v}_k$.

To make CBS learning more effective by generating more similarity features, we can use multiple document space representations or feature vectors to represent each document d_k and also employ multiple similarity measures. The detailed technique is as follows.

Instead of using only one *ds*-vector $\mathbf{x}_k$ to represent a document d_k, we use a set R_k of *ds*-vectors $R_k = \{\boldsymbol{x}_1^k, \boldsymbol{x}_2^k, \ldots, \boldsymbol{x}_p^k\}$, where each *ds*-vector $\boldsymbol{x}_j^k$ denotes one document space representation of d_k, e.g., unigram or bigram. Due to multiple representations, we have multiple centers also for the positive class, $C = \{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_p\}$, where each $\mathbf{c}_j$ corresponds to one document space representation . We use the Rocchio method in information retrieval [19, 23] to compute $\boldsymbol{c}_j$ (a vector) using the corresponding *ds*-vectors of all positive and negative training documents.

$$\mathbf{c}_j = \frac{\alpha}{|D_+|} \sum_{d_k \in D_+} \frac{\mathbf{x}_j^k}{\|\mathbf{x}_j^k\|} - \frac{\beta}{|D - D_+|} \sum_{d_k \in D - D_+} \frac{\mathbf{x}_j^k}{\|\mathbf{x}_j^k\|}$$

where D_+ is the set of positive documents and $|.|$ is the size function. α and β are parameters. It is reported in [4] that using tf-idf representation, $\alpha = 16$ and $\beta = 4$ usually work quite well. The subtraction is applied to reduce the influence of those terms that are not discriminative (i.e., terms appearing in both classes).

Based on R_k for document d_k (either in the training and testing set) and the computed set of centers C (computed using only the training data), we can transform a document d_k from its document space representations R_k to a center-based similarity vector $\mathbf{cbs\text{-}v}_k$ by applying a similarity function Sim on each element $\mathbf{x}_j^k$ of R_k and its corresponding center $\mathbf{c}_j$ in C.

$$\mathbf{cbs\text{-}v}_k = Sim(R_k, C)$$

Sim can contain a set of similarity measures. Each measure m is applied to p document representations in R_k and their corresponding centers in C to generate p similarity features (*cbs*-features) in $\mathbf{cbs\text{-}v}_k$.

For *ds*-features, unigrams and bigrams with tf-idf weighting were used as two document representations. The similarity measures are the five ones described in [13], which produce 10 CBS features to represent a document in the CBS space. Based on the CBS space representation, we can run SVM to produce a CBS classifier h_y.

We now briefly explain why CBS learning gives a good estimate to S_o. Due to using similarities as features, CBS learning generates a boundary to separate the positive and negative training data in the similarity space. As a similarity has no direction (or it covers all directions), the boundary in the similarity space is essentially a "ball" encompassing the target/positive class training data in the original document space. The "ball" is an estimate of S_o based on those similarity measures. The main assumption by the CBS learning is that the target class data should be unimodal in order for the "ball" not to cover too much open space.

5. Evaluation

In this section, we evaluate our proposed system and compare it with extensive baselines in terms of both classification results and computational speed.

5.1 Datasets

To evaluate the proposed cumulative learning method, we need a fully labeled document collection with a large number of classes. One kind of dataset that naturally comes with a large number of classes is product reviews, which contains reviews for different product categories. Note that although we use this product reviews collection, we do not perform sentiment classification. Instead, we still perform the traditional topic based classification. That is, given a review, the system has to decide what type of product the review is about. We use Amazon product reviews of 100 domains (or types of products) that were used in [8] as one dataset. Each domain contains 1000 reviews. We also perform evaluation using another publicly available dataset 20-newsgroup. For 20-newsgroup, we use the "18828" version, which contains 20 non-overlapping classes, in total 18828 text documents with no duplicates. For each class/domain in both datasets, we randomly keep 70% of the documents for training and the rest 30% for testing.

For experiments in this paper, we do not perform any over-sampling or under-sampling for all methods for two reasons. First, as we will discuss later, our experiments have multiple settings that use different number of domains, it is hard to select the optimal sampling number every time, and it is also not the focus of our paper. Secondly, since this strategy applies to all the methods, we do not bias against any one.

5.2 Baselines

Our supervised learning baselines can be classified into two categories depending on the strategy they use given new classes of documents. All the supervised learning baselines discussed below except *CL-1-vs-rest-SVM* are based on rebuilding strategy. *CL-1-vs-rest-SVM* is a variation of our proposed *CL-cbsSVM*, which is able to cumulatively update the system given a new class of documents. For more complete evaluation, apart from supervised learning baseline methods, we also include a semi-supervised learning method, called *Exploratory-EM*, as a baseline because it allows new classes to be created during the EM iterations. All the baseline methods are listed below.

1-vs-Rest multiclass SVM (*1-vs-rest-SVM*). This is the standard 1-vs-rest multiclass SVM with Platt Probability Estimation [27], which only supports rebuilding strategy. It works in the same way as *cbsSVM* [14] except that it uses the document space classification. This baseline is based on the linear SVM with probabilistic outputs in LIBSVM (version 3.20) [7]. Similar to *cbsSVM*, for open world classification, decision threshold $\theta = 0.5$ is used for detecting documents from new classes (similar to that in Section 3.2). We use the linear kernel as many researchers have shown that linear SVM performs the best for text classification [19]. We also tried other kernels, but they were poorer.

1-vs-Set Machine (*1-vs-set-linear*). This is the 1-vs-Set Machine in [31], which only supports rebuilding strategy. We use all the default parameter settings in the original paper. That is, the near and far plane pressures are set at $p_A = 1.6$ and $p_\Omega = 4$ respectively; regularization constant $\lambda_r = 1$ and no explicit hard constraints are used on the training error ($\alpha = 0, \beta = 1$).

W-SVM (*wsvm-linear* and *wsvm-rbf*). These two baselines combine RBF one-class SVM with binary SVM [32]. Both linear kernel and RBF kernel are tested. For thresholding the output, two parameters δ_τ and δ_R are required. We set $\delta_\tau = 0.001$, which is used to adjust what data the one-class SVM considers to be remotely related. δ_R is a required decision threshold with the same effect as our θ, which is not only for W-SVM, but also for the next baselines (P_I-SVM). Two ways of setting δ_R were suggested by the paper authors. We set it as the prior probability of the number of unseen classes during evaluation (testing). An alternative way is to set it based on an openness score computed using the number of training and testing classes. We tried both methods and found the former gave better results.

P_I-SVM (*P_I-svm-linear* and *P_I-svm-rbf*). These two baselines are from [18], which estimate the probability of inclusion based on output of binary SVMs with two kernels. As stated above, the threshold δ is set as the prior probability of the number of unseen classes in test. As both W-SVM and P_I-SVM need pre-trained 1-vs-rest SVM and the step of calibrating SVM scores by fitting WeiBull distributions, both of these baselines only support rebuilding strategy given new classes of documents.

Cumulative Learning with 1-vs-Rest SVM (*CL-1-vs-rest-SVM*). Intuitively, the choice of the underlying learner should affect cumulative learning in both classification result and running time. As our proposed cumulative learning process is independent of the learner used in building the classifier, instead of using *cbsSVM* as the underlying learner, we apply *1-vs-rest-SVM* in cumulative learning process (see Section 3.1) as another baseline. This is the only supervised learning baseline that supports cumulative update of the model without rebuilding the system from scratch given a new class of documents.

Exploratory Seeded K-Means (*Exploratory-EM*). In [11], three well-known multiclass semi-supervised learning methods were extended under the exploratory EM framework. We compare with exploratory Seeded K-Means due to its superior performance on 20newsgroup dataset. We also applied the criteria that work the best in the original paper for creating new classes and for model selection, i.e., the MinMax criterion and the AICc criterion. Note that *Exploratory-EM* works in the semi-supervised setting and uses both the training and test data as labeled and unlabeled data in learning. The algorithm supports two modes. The "explore" mode allows new classes to be created while the "semisup" mode does not. As more than one new class can be introduced during learning in the "explore" mode, for comparison we lump together all instances assigned to new classes as being rejected (unknown). In the experiments, we set the max number of iterations to be 50. Little changes in results are shown after 50 iterations.

	m =33%	66%	100%	33%	66%	100%	33%	66%	100%	33%	66%	100%
1-vs-rest-SVM	0.498	0.501	0.568	0.442	0.490	0.541	0.460	0.444	0.418	*0.652*	0.714	0.808
cbsSVM	**0.580**	**0.632**	**0.639**	**0.546**	**0.581**	**0.619**	**0.579**	**0.565**	**0.569**	**0.662**	**0.728**	**0.835**
CL-cbsSVM	*0.549*	*0.610*	*0.623*	*0.511*	0.574	0.616	*0.536*	0.552	0.549	0.644	*0.716*	0.820
CL-1-vs-rest-SVM	0.352	0.511	0.472	0.488	0.440	0.424	0.352	0.373	0.394	0.417	0.632	0.713
1-vs-set-linear	0.437	0.496	0.334	0.379	0.499	0.534	0.379	0.463	0.290	0.620	0.529	0.606
wsvm-linear	0.506	0.537	0.335	0.454	0.535	0.547	0.465	0.499	0.309	0.597	0.606	0.710
wsvm-rbf	0.347	0.382	0.398	0.278	0.357	0.544	0.264	0.289	0.095	0.417	0.643	0.812
P_i-svm-linear	0.507	0.539	0.337	0.454	0.536	0.550	0.465	0.499	0.303	0.598	0.608	0.712
P_i-svm-rbf	0.407	0.595	0.409	0.388	*0.576*	0.603	0.389	*0.562*	0.310	0.435	0.715	0.806
ExploratoryEM	0.419	0.523	0.618	0.366	0.514	0.576	0.377	0.480	0.538	0.559	0.690	*0.823*
	(a) amazon (n=50)			(b) amazon (n=75)			(c) amazon (n=100)			(d) 20newsgroup (n=20)		

Table 1. Macro-F1 scores for cumulative learning under open world classification

All documents use tf-idf term weighting scheme with no feature selection. Source code for baselines such as 1-vs-Set Machine[1], W-SVM and P_I-SVM[2], and Exploratory learning[3], is provided by the authors of their original papers.

One thing to note is that we did not include any 1-vs-1 SVM based multiclass classification methods as baselines. This is because although a 1-vs-1 SVM technique for multiclass classification can support learning cumulatively by adding t new 1-vs-1 classifiers for the arrival of the $(t+1)^{th}$ class, none of the existing such methods can support open world classification.

5.3 Experimental Settings

Following that in [11, 18], we conduct open world cross-validation style evaluation, holding out some classes in training and mixing them back during testing, and varying the number of training and test classes. Since for a given dataset, the number (percentage) of training classes m and the number of test classes n can vary, there are many ways to generate a train-test partition. We report our results using 5 random train-test partitions for each combination of m and n. In particular, we vary the number of test classes for Amazon ($n = 50, 75, 100$), and for 20-newsgroup ($n = 20$). For each of these choices on the number of test classes, we also select m = 33%, 66% and 100% of the number of test classes for training. Varying the ratio of the number of training classes to test classes is to test the robustness of different systems in handling different "openness" of the problem.

When m = 100% of test classes are used for training, the problem reduces to the closed world classification. As most of our baselines such as those based on W-SVM and P_I-SVM all use the prior knowledge to set decision threshold $\delta_R = 0$ in the closed world setting, for fair comparison, we also set the threshold $\theta = 0$ for methods *1-vs-rest-SVM*, *CL-1-vs-rest-SVM*, *cbsSVM* and *CL-cbsSVM*. By doing this, we always assign a known class label to a test instance. For *Exploratory-EM*, we use the provided "semisup" mode instead of "explore" mode, which allows no new classes to be introduced in learning.

For methods that support our proposed cumulative update (*CL-cbsSVM* and *CL-1-vs-rest-SVM*), we build a k-class classifier system starting from only 2 classes and cumulatively add new classes to the system one at a time till k classes. We set $\lambda_{sim} = 2\%$ for all methods, as it gives the best results, and we will discuss its effect in section 5.4.2. For methods that use rebuilding strategy (*1-vs-rest-SVM*, *1-vs-set-linear*, *wsvm-linear*, *wsvm-rbf*, *Pi-svm-linea*, and *Pi-svm-rbf*, *cbsSVM*), instead of simulating the whole process of building classifiers starting from 2 classes till k classes by rebuilding the system over and over again, we only build a k-class classifier once using all the k classes to simulate what happens when the k^{th} class arrives.

For all the methods that use the RBF kernel, the parameters are tuned via cross validation on the training data, yielding ($C = 5, \gamma = 0.2$) for Amazon and ($C = 10, \gamma = 0.5$) for 20-newsgroup.

5.4 Experimental Results and Comparison

In this section, we first show the classification results of all the methods discussed in this paper on both the open world and closed world classification tasks. We then conduct running time analysis in section 5.4.2 to compare their efficiency. Finally, we perform qualitative analysis of the proposed cumulative learning.

5.4.1 Classification Results

In order to compare the classification results of different systems, for each train-test partition, we first compute precision, recall and F1 score for each class and then macro-average the results across all classes. Final results are given by averaging the results of 5 random train-test partitions. We only show F1 scores in the paper.

We show the results of different methods on open world classification in Table 1, which has four sets of results. More specifically, from left to right, we show results of different methods on Amazon data when the number of test classes $n = 50, 75, 100$, and 20newsgroup data when $n = 20$. Within each sub-table, different columns list results of different methods when different proportions of test classes are used for training. From Table 1, we can clearly see that *cbsSVM* performs the best in all settings. Even when 100% of the test classes are used for training (the traditional closed world classification), *cbsSVM* still performs the best in all settings. Note that it does not use cumulative update of the system given new classes of documents. That is, it is based on the re-building strategy.

We also notice that the proposed *CL-cbsSVM* (which is the cumulative version) almost always gives the second best results except that it loses to *P_i-svm-rbf* in one setting ($n = 75$, $m = 66\%$), but better than all other methods based on rebuilding strategy. It is not surprising that *CL-cbsSVM* loses to *cbsSVM*,

[1] https://github.com/Vastlab/liblinear.git
[2] https://github.com/ljain2/libsvm-openset
[3] http://www.cs.cmu.edu/~bbd/ExploreEM_package.zip

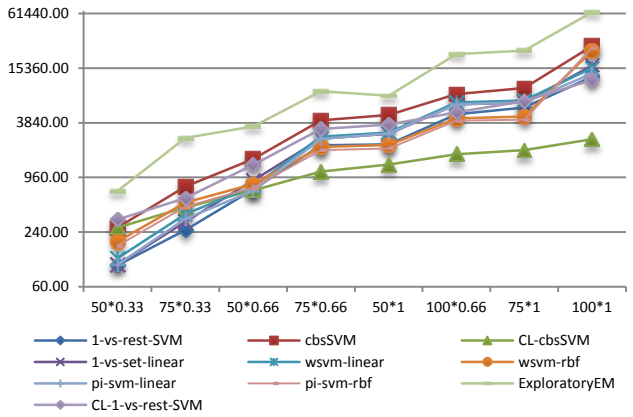

Figure 2. Running Time Summary

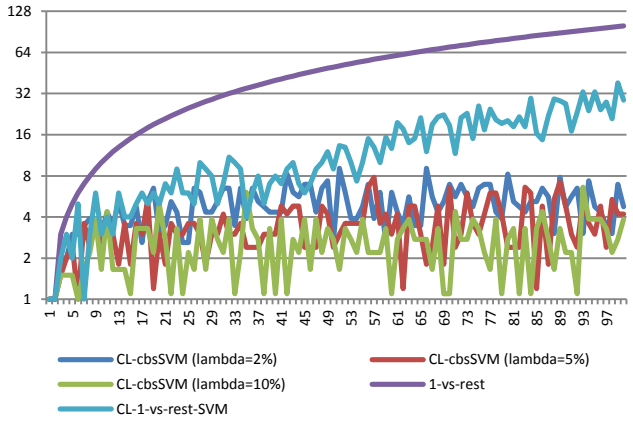

Figure 3. Number of Classifiers Trained Each Time

CL-cbsSVM					CL-1-vs-rest-SVM				
Diaper	***Clothing***	***Headphone***	***Lamp***	***Laptop***	***Diaper***	***Clothing***	***Headphone***	***Lamp***	***Laptop***
Baby	Baby	CDPlayer	Kindle	HardDrive	Baby	Baby	DVDPlayer	Home Improvement	HardDrive
Clothing	ArtsCrafts Sewing	CarStereo	Books	Lamp	Care	Automotive	CDPlayer	Kindle	Computer
	Care	Amplifier		Battery	DVDPlayer	Care	CarStereo	Automotive	Clothing
	Beauty	Care		Computer	Clothing	Beauty	Amplifier	Industrial Scientific	Automotive
				Clothing	ArtsCrafts Sewing	ArtsCrafts Sewing	Clothing	DVDPlayer	Home Improvement
				GPS	Automotive	Battery	Care	Clothing	Battery
					Beauty	CarStereo	Gloves	Battery	Car Stereo
					CDPlayer	CDPlayer	Golf	CDPlayer	DVDPlayer
					BluRay Player	Camcorder (2 more)	GPS (1 more)	Care (9 more)	Kindle (11 more)

Table 2. Selected Negative Training Classes by *CL-1-vs-rest-SVM* and *CL-cbsSVM*

because *cbsSVM* uses all the classes when building the classifier while *CL-cbsSVM* only rely on a small subset of previous classes. This leads to dramatic efficiency increase over *cbsSVM* as will be shown in Section 5.4.2. In contrast to *CL-cbsSVM*, the baseline method *CL-1-vs-rest-SVM*, which also supports learning cumulatively, performs poorly in both open world and closed world classification tasks. The main reason is that its underlying learner SVM learns in the document space. Due to this reason, SVM is not suitable in handling unseen classes, identifying similar domains, or learning effectively given similar classes during the cumulative learning process.

We also notice that *Exploratory-EM* gives one of the best results in closed world classification tasks, but doesn't perform very well in open world situations. We believe this is because *Exploratory-EM* uses test data in learning, so it performs well in traditional semi-supervised learning tasks. But in open world classification tasks, as stated in [11], *Exploratory-EM* only focuses on the domains in which seed (training) documents are provided, which is not the case in our setting.

5.4.2 Running Time Analysis

Apart from classification results, we are also interested in evaluating the efficiency of each method. For comparison, we measure the time it takes for different systems to rebuild or cumulatively update when the last new class is presented in each setting. For methods based on the rebuilding strategy, this is equivalent to the total training time because we only build the system once using all the training classes.

In this paper, all experiments are conducted using a single thread on a single 64-bit Windows server with Intel Xeon X5650 2.67GHz CPU. At the same time, we are also aware that some of the experimented methods can be easily parallelized. For example, a single classifier can be trained on a different machine for *1-vs-rest-SVM*, *1-vs-set-linear* and *cbsSVM*. And for methods based on the proposed cumulative learning process *CL-cbsSVM* and *CL-1-vs-rest-SVM*, both detecting similar classes and building/updating existing classifiers can be sent to different machines to speed up the overall running time. However, we plan to leave this part of work to the future.

For all the methods except *Exploratory-EM*, we implemented data preprocessing and transformation in Java, and called LIBSVM [7] or its extension packages for training. Summary of the running time of different methods is shown in Figure 2. The x-axis indicates experimental settings. For example, 50 * 0.33 indicates the setting in which 50 * 0.33 (=16) classes are used to train, and 50 classes for testing. The reason we show it this way is because *Exploratory-EM* is a semi-supervised method, the number of test domains also affects its running time. The y-axis indicates running time (in seconds) to rebuild/update the system when the last domain (50 * 0.33 = 16th) arrives. Due to the wide range of running time of different algorithms, the y-axis (seconds) is in $\log_2$ scale. However, note that two reasons made the running time of *Exploratory-EM* algorithm not very comparable. Firstly, the Exploratory Learning package was written in Matlab. Secondly, it's a semi-supervised method, which uses test data during learning.

From Figure 2, we are able to make several interesting observations. Firstly, apart from *Exploratory-EM*, *cbsSVM* takes longer time to run compared to other 1-vs-rest methods, which is due to the extra data preprocessing and transformation overhead and the use of RBF kernel. Secondly, our proposed *CL-cbsSVM* takes the shortest time to update especially when the number of classes is big. Due to the data transformation overhead, it is slightly slower than some other baselines when the number of classes is small. Thirdly, although also based on the updating strategy, running speed of *CL-1-vs-rest-SVM* is dramatically slower than that of *CL-cbsSVM*. This is because *CL-1-vs-rest-SVM* uses SVM as the underlying learner and is not good at identifying new classes. Thus many existing classifiers misclassify many instances from the new class and get retrained. As the number of classes in the system increases over time, this effect is amplified further. Lastly, most of other algorithms based on 1-vs-rest SVM (*1-vs-rest-SVM, 1-vs-set-linear, wsvm-linear, wsvm-rbf, Pi-svm-linear* and *Pi-svm-rbf*) have similar running time and trend with the increase on the number of training classes/domains.

Although the running time of different methods is affected by many factors, another way of comparing the efficiency of different approaches is to look at the number of classifiers they build/update each time a new class arrives starting from two classes to *k* classes. In Figure 3, we show results for *CL-1-vs-rest-SVM*, *CL-cbsSVM* as well as all methods based on the rebuilding strategy. The y-axis indicates the number of classifiers that were built/updated (in log2 scale) and x-axis is the arrival of the i^{th} class. As all the methods based on the rebuilding strategy always build t classifiers when the t^{th} new class is presented to the system, it is thus the upper bound for *CL-1-vs-rest-SVM* and *CL-cbsSVM*. To illustrate the effect of λ_{sim} parameter, we vary the parameter $\lambda_{sim} = 10\%, 5\%, 2\%$ for both systems. But for the simplicity of the plot, we only show $\lambda_{sim} = 2\%$ for *CL-1-vs-rest-SVM*, as both methods show similar trends by varying λ_{sim}. Since the sequence of arrival of new classes affects the results of our method, we average the numbers across 5 runs.

From Figure 3, we can see that *CL-1-vs-rest-SVM* clearly builds more classifiers than *CL-cbsSVM* does when λ_{sim} is set at 2% for both systems. By comparing different λ_{sim} values for *CL-cbsSVM*, we see that on average smaller λ_{sim} leads to more classifiers being updated. This is intuitive because smaller λ_{sim} indicates being stricter on if an old classifier should be updated, while at the same time, smaller λ_{sim} gives slightly better accuracy in classification, which is also intuitive.

5.4.3 Qualitative Analysis of Cumulative Learning

One way to gain more insight into our proposed cumulative learning system *CL-cbsSVM* is by looking at what existing classes are selected as the negative training data during its learning process in building/updating classifiers when new classes arrive. Intuitively, an effective and efficient learning algorithm should be able to select the minimal number of closely relevant domains instead of lots of irrelevant ones. For comparison, we also show the results from *CL-1-vs-rest-SVM* with the same class arrival order. Table 2 lists the negative classes picked by both methods for the 26^{th} till the 30^{th} new class when building the system cumulatively on the Amazon data. The second row shows the names of the new classes, and each column shows the chosen similar classes by each system. The reason we pick these five positions is because there are enough existing classes for a system to make mistakes, and not too many so that we can still list them in the paper. In fact, *CL-1-vs-rest-SVM* still picked too many classes beyond what we can enumerate in the paper. We also tried to manually identify those picked classes that we feel are unrelated, and they are marked in red.

From the results shown in Table 2, we can easily tell that not only the number of negative training classes selected by *CL-cbsSVM* is much smaller, they are also more relevant than those picked by *CL-1-vs-rest-SVM*.

6. Conclusion

This paper proposed the new task of cumulative learning as a special form of supervised lifelong machine learning. Two unique challenges in cumulative learning are identified and presented, namely, the ability to detect data from unseen classes in the test set and the ability to selectively update the existing classification model without requiring rebuilding the whole system from scratch. We also proposed a cumulative learning strategy, which we believe is similar to human concept learning. It only requires updating part of the existing classification model whenever a new class of data arrives and needs to be covered by the classification model. As time goes by, the system keeps learning more and more in an efficient manner and becomes more and more knowledgeable. To deal with the first problem of open world classification, we adopted a recent solution based on center-based similarity space (CBS) learning, which is able to balance open space risk and empirical risk during training. Through extensive experiments by building classifiers for up to 100 consecutive classes, and comparing with strong baselines, we demonstrated the effectiveness and efficiency of the proposed approach.

7. ACKNOWLEDGMENTS

This work was supported in part by a grant from National Science Foundation (NSF) under grant no. IIS-1407927, a NCI grant under grant no. R01CA192240, and a gift from Bosch.

8. REFERENCES

[1] A. Blum. On-line algorithms in machine learning. In Proceedings of the Workshop on On-Line Algorithms, Dagstuhl, pages 306-325. 1996.

[2] C. Bravo, J. L. Lobato, R. Weber, and G. L'Huillier. A hybrid system for probability estimation in multiclass problems combining svms and neural networks. In HIS '08: Proceedings of the 2008 8thInternational Conference on Hybrid Intelligent Systems, pages 649-654, 2008.

[3] E. Brunskill and L. Li. Pac-inspired option discovery in lifelong reinforcement learning. In T. Jebara and E. P. Xing, editors, Proceedings of the 31st International Conference on Machine Learning (ICML-14), pages 316–324. JMLR Workshop and Conference Proceedings, 2014.

[4] C. Buckley, G. Salton, and J. Allan. The effect of adding relevance information in a relevance feedback environment. In Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '94, pages 292–300, 1994.

[5] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka, and T. M. Mitchell. Toward an architecture for never-ending language learning. In AAAI, 2010.

[6] G. Cauwenberghs and T. Poggio. Incremental and decremental support vector machine learning, 2000.

[7] C.-C. Chang and C.-J. Lin. Libsvm: A library for support vector machines. ACM Trans. Intell. Syst. Technol., 2(3):27:1–27:27, May 2011.

[8] Z. Chen and B. Liu. Topic modeling using topics from many domains, lifelong learning and big data. In ICML, pages 703–711, 2014.

[9] Z. Chen, N. Ma and B. Liu. Lifelong Learning for Sentiment Classification. Proceedings of the 53st Annual Meeting of the Association for Computational Linguistics (ACL-2015, short paper), 26-31, July 2015, Beijing, China.

[10] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. Journal Of Machine Learning Research, 7:551–585, 2006.

[11] B. Dalvi, W. W. Cohen, and J. Callan. Exploratory learning. In ECML, 2013.

[12] K.-B. Duan and S. S. Keerthi. Which is the best multiclass svm method? An empirical study. In Proceedings of the 6th International Conference on Multiple Classifier Systems, MCS'05, pages 278–285, Berlin, Heidelberg, 2005.

[13] G. Fei and B. Liu. Social media text classification under negative covariate shift. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2347–2356, Lisbon, Portugal.

[14] G. Fei and B. Liu. Breaking the Closed World Assumption in Text Classification. In Proceedings of the 2016 NAACL.

[15] M. Fink, S. Shalev-Shwartz, Y. Singer, and S. Ullman. Online multiclass learning by interclass hypothesis sharing. In Proceedings of the 23rd International Conference on Machine Learning, ICML '06, pages 313–320, 2006.

[16] V. J. Hodge and J. Austin. A survey of outlier detection methodologies. Artificial Intelligence Review, 22:2004, 2004.

[17] T.-K. Huang, R. C. Weng, and C.-J. Lin. Generalized Bradley-Terry models and multi-class probability estimates. J. Mach. Learn. Res., 7:85–115, Dec. 2006.

[18] L. P. Jain, W. J. Scheirer, and T. E. Boult. Multi-class open set recognition using probability of inclusion. In The European Conference on Computer Vision (ECCV), September 2014.

[19] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In Proceedings of the 10th European Conference on Machine Learning, ECML '98, pages 137–142, 1998.

[20] S. M. Kakade, S. Shalev-Shwartz, and A. Tewari. Efficient bandit algorithms for online multiclass prediction. In Proceedings of the 25th International Conference on Machine Learning, ICML '08, pages 440–447, 2008.

[21] J. Kang, J. Ma, and Y. Liu. Transfer topic modeling with ease and scalability. In Proceedings of SDM, pages 564–575. 2012

[22] S. S. Khan and M. G. Madden. One-class classification: Taxonomy of study and review of techniques. CoRR, abs/1312.0049, 2013.

[23] C. D. Manning, P. Raghavan, and H. SchAijtze. Introduction to Information Retrieval. Cambridge University Press, 2008.

[24] M. Markou and S. Singh. Novelty detection: A review- part 1: Statistical approaches. Signal Processing, 83:2003, 2003.

[25] S. J. Pan and Q. Yang. A survey on transfer learning.

[26] A. Pentina and C. H. Lampert. A pac-bayesian bound for lifelong learning. In International Conference on Machine Learning (ICML), 2014.

[27] J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In ADVANCES IN LARGE MARGIN CLASSIFIERS, pages 61–74. MIT Press, 1999.

[28] J. J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, The Smart retrieval system - experiments in automatic document processing, pages 313–323. Englewood Cliffs, NJ: Prentice-Hall, 1971.

[29] P. Ruvolo and E. Eaton. Active task selection for lifelong machine learning. In Proceedings of the 27th AAAI Conference on Artificial Intelligence (AAAI-13), July 2013.

[30] P. Ruvolo and E. Eaton. Ella: An efficient lifelong learning algorithm. In ICML (1), volume 28 of JMLR Proceedings, pages 507–515. JMLR.org, 2013.

[31] W. J. Scheirer and T. E. Boult. Towards open set recognition.

[32] W. J. Scheirer, L. P. Jain, and T. E. Boult. Probability models for open set recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI), 36, 11/ 2014.

[33] B. Schölkopf, J. C. Platt, J. Shawe-taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution, 1999.

[34] N. Shackel. Bertranda's paradox and the principle of indifference. Philosophy of Science, 74(2): 150–175, 2007.

[35] D. L. Silver, Q. Yang, and L. Li. Lifelong machine learning systems: Beyond learning algorithms. In AAAI Spring Symposium: Lifelong Machine Learning, volume SS-13-05 of AAAI Technical Report. AAAI, 2013.

[36] D. L. Silver and R. Poirier. Sequential consolidation of learned task knowledge. In Proceedings of the17th Conference of the Canadian Society for Computational Studies of Intelligence, Canadian AI 2004, Canada, May 17-19, 2004.

[37] D. M. J. Tax and R. P. W. Duin. Support vector domain description. Pattern Recognition Letters, 20:1191–1199, 1999.

[38] S. Thrun. Is learning the n-th thing any easier than learning the first? In NIPS, pages 640–646, 1996.

[39] S. Thrun and T. M. Mitchell. Lifelong robot learning. Technical report, Robotics and Autonomous Systems, 1993.

[40] T. Xiao, J. Zhang, K. Yang, Y. Peng, and Z. Zhang. Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In ACM Multimedia, 2014.

[41] B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates, 2002.

[42] P. Zhao, S. C. H. Hoi, and R. Jin. Double updating online learning. Journal of Machine Learning Research, 12:1587–1615, 2011.