Stochastic Gradient Descent and the Prediction of MeSH for PubMed Records

W. John Wilbur MD, PhD and Won Kim PhD
National Center for Biotechnology Information, National Library of Medicine, National Institutes of Health, Bethesda, MD, U.S.A.

Abstract

Stochastic Gradient Descent (SGD) has gained popularity for solving large scale supervised machine learning problems. It provides a rapid method for minimizing a number of loss functions and is applicable to Support Vector Machine (SVM) and Logistic optimizations. However SGD does not provide a convenient stopping criterion. Generally an optimal number of iterations over the data may be determined using held out data. Here we compare stopping predictions based on held out data with simply stopping at a fixed number of iterations and show that the latter works as well as the former for a number of commonly studied text classification problems. In particular fixed stopping works well for MeSH® predictions on PubMed® records. We also surveyed the published algorithms for SVM learning on large data sets, and chose three for comparison: PROBE, SVMperf, and Liblinear and compared them with SGD with a fixed number of iterations. We find SGD with a fixed number of iterations performs as well as these alternative methods and is much faster to compute. As an application we made SGD-SVM predictions for all MeSH terms and used the Pool Adjacent Violators (PAV) algorithm to convert these predictions to probabilities. Such probabilistic predictions lead to ranked MeSH term predictions superior to previously published results on two test sets.

Introduction

The National Library of Medicine (NLM) produces the PubMed database of biomedical journal article citation records consisting of title, abstract (where available) and appropriate metadata. This data base currently contains over 23 million records and is growing at about 60,000 records per month. Most of these records have approximately a dozen key terms assigned to them from a controlled vocabulary known as Medical Subject Headings (MeSH). There are over 27,000 MeSH terms from which these key terms or index terms are assigned by humans. This involves significant human effort and expense. It is then natural that efforts would be made to mechanize some of this work. Such efforts have been of two types. First, there has been an ongoing effort to develop a system called the Medical Text Indexer (MTI) to predict MeSH assignments as suggestions for MeSH indexers at the NLM[1-9]. Second, there have been a number of investigations of machine learning methods and how they might be applied to the MeSH indexing problem in a more abstract setting with the purpose of understanding and comparing different approaches to this problem[6, 10-15]. The work we report here is of this latter type. We examine the Stochastic Gradient Descent (SGD) method for solving Support Vector Machines (SVMs)[16, 17] and develop an approach which we believe has broad applicability but is especially attractive for solving very large SVM problems. We show how to use this approach to improve MeSH suggestions over prior published work.

SGD has proved to be a very effective method of training machine learning algorithms[16, 17]. It has generally been found to confer a significant decrease in training time without sacrificing accuracy[17-19]. SGD can be applied to standard convex loss functions with regularization terms with good effect, but Zhang[17] suggested using SGD without the usual regularization term and performing the regularization with early stopping. This has become a widely practiced approach[20-22] and is implemented by dividing the training set into disjoint pieces consisting of a new training set and a validation set. Then on each training pass through the new training set one tests on the validation set and that number of iterations over the training data which is found to be optimal is recorded. One then trains on the original training set with this number of iterations and evaluates the results on the test set to rate the method. One of our contributions in this work is to show that on a number of text classification problems one can implement early stopping by just using a constant number of iterations and obtain the same performance as one obtains using the validation set approach.

In particular we find that SGD with a fixed number of eight iterations over the training data provides MeSH classification results as good as early stopping based on held out data and as good as several other popular methods which take much longer to train. Given the efficiency of the method we are able to apply it to each MeSH heading to make predictions. It would then be possible to use the SVM scores of all the MeSH terms for a given document to rank the predictions for that document. However, different MeSH terms have different frequencies in the training data and this leads to classifier scores that are not directly comparable, i.e., one does not obtain the best result with such a ranking approach based on raw scores. This leads us to take a slightly different approach. We first divide the training data into two equal halves. For a given MeSH term we train a classifier on each half. We then apply the Pool Adjacent Violators (PAV) algorithm[23-26] to each classifier's scoring of the half of the training data on which it was not trained. The PAV algorithm converts raw scores to probabilities. As a consequence, for any previously unseen test document, we can apply the classifiers to obtain two raw scores and use the PAV models to convert these raw scores to probabilities. We average the two probabilities to obtain a single probability estimate for that MeSH term for the test document. Since probabilities are optimally comparable we use the resulting probabilities over all MeSH terms to make ranked MeSH term predictions. We find such predictions superior to predictions obtained by raw score ranking and also superior to previous published predictions.

Methods

The SGD Algorithm with Early Stopping Assume we are given N training points $\{(x_i, y_i)\}_{i=1}^{N}$ randomly and independently sampled from the same source distribution, where for each i, $x_i \in R^n$ and $y_i \in \{-1,1\}$. The objective is to learn a weight vector $w \in R^n$ so that the function

$$f_w(x) = \operatorname{sgn}(w \cdot x) \tag{1}$$

has a high probability of agreeing with y if (x, y) is randomly sampled from the same source as the training data. Here the function sgn is known as the sign function and has the definition

$$\operatorname{sgn}(x) = \begin{cases} 1, & x > 0 \\ 0, & x = 0 \\ -1, & x < 0 \end{cases} . \tag{2}$$

Then pseudocode for the SGD algorithm without regularization, but with early stopping is as follows.

```
SGD without regularization
Input: training data $\{(x_i, y_i)\}_{i=1}^{N}$; learning rate $\lambda > 0$.
Initialize: $w_0 = 0$; $t = 0$.
A. Randomly sample $j,\ 1 \le j \le N$
B. If $y_j(w_t \cdot x_j) < 1$ set $w_{t+1} = w_t + \lambda y_j x_j$ and $t = t + 1$.
C. if stopping criterion satisfied return $w_t$ else return to A.
```

Several issues require comment here. First, we do not deal with a threshold independent of the vectors x_i but assume that all vectors have a common added dimension which is also present in w and functions as threshold. Second, we do not randomly sample integers from the interval $[1, N]$ as in step A of the algorithm. Instead we do the training in rounds or passes over the training data and before each pass we re-randomize the order in which the integers in $[1, N]$ are visited. Third, we use a learning rate $\lambda = 0.002$. On all three of these points we are simply following[17]. Our pseudocode does not define the stopping criterion. Generally cross validation or held out data is used to determine a stopping point, but our purpose is to compare that approach to simply stopping at a fixed number of passes over the data without doing any cross validation.

Method of SGD Experiments By holding out training data, we may experimentally determine the optimal number of iterations of SGD for SVM with early stopping (L1 loss Minimization without regularization, hereafter we call it SGD-SVM)[20-22]. Let $D = \{(x_n, y_n), n = 1, \cdots, N\}$ be our data set, where x_n is a vector representing the attributes of the nth instance and y_n is the binary class value of x_n. We randomly split the data into J equal parts $D_1, D_2 \cdots D_J$. Let D_j and $D^{(-j)} = D - D_j$ be the test and training sets for the jth fold of a J fold cross validation. For the jth fold we again randomly split the training set $D^{(-j)}$ into K equal parts $D_1^{(-j)} \cdots D_K^{(-j)}$. Defining $D_K^{(-j)}$ to be the held out or validation set, we may determine the optimal number of iterations of SVM-SGD. We train SGD-SVM on $D^{(-j)} - D_K^{(-j)}$ and let $Iter_j$ be the optimal iteration number for SGD-SVM evaluated on $D_K^{(-j)}$. We then apply SVM-SGD learning with $Iter_j$ iterations on $D^{(-j)}$, and measure the success of this learning on D^j. Note that random splits are done in such a way as to keep the number of negative records and the number of positive records the same or as close to the same as possible over all splits. The J different test results from the J different folds are combined by macro-averaging , which we believe is appropriate when all J experiments are very similar. We report mean average precision (MAP) and break even (BE) values[27] and macro-averaging means we compute the measure for each fold and then take a simple average over all folds for the final value reported. The break even value is sometimes also known as R-precision[27].

When we do the experiment without using held out data the protocol is simpler. As before, we randomly split the data into J equal parts $D_1, D_2 \cdots D_J$ and let D_j and $D^{(-j)} = D - D_j$ be the test and training sets for the jth fold of a J fold cross validation. But now $Iter_j$ is a fixed number for all j. One may ask how do we decide what this number should be. Our answer is empirical. We have observed what works well on the data from much experience based on cross validations and other experiments where we simply run the learning algorithm for a number of iterations and watch the performance. Our conclusions are surprisingly simple. For all the MeSH experiments we use 8 iterations and for all the other classification problems we study in this work we find 9 iterations give good results. We suspect this difference is due to the much larger size of the training data for MeSH than for any of the other problems we study.

Data Sets In addition to studying classification for MeSH assignment, we also study six smaller textual databases, each of which is associated with one or more classification problems. The databases and their sizes are listed in Table 1. Other than the PubMed database these sources were prepared previously by the authors and documented in[28] and to save space we refer the reader to this source for details. The PubMed database used is a March 2013 copy.

Table 1. Corpora used in this study. Given for each corpus is the number of defined sub-problems, the number of examples, and the number of features.

Database	classes	Examples	Features
REBASE	2	102,997	2,032,075
MDRDataset	3	620,119	738,104
Newsgroups	20	20,000	98,587
IndustrySectors	104	9,555	55,056
WebKB	7	8,280	69,911
Reuters	10	27,578	46,623
PubMed	27	22,411,501	77,040,540

PubMed We study a March 2013 version of the PubMed database. As noted above, there are over 23 million records in PubMed representing as many journal articles mostly on diverse biomedical topics. More detail can be obtained online[29]. Each record in the database is represented by features consisting of the words and two word colocations contained in the title and abstract. A stoplist of function words are not allowed in these features, but no stemming is done. We divide this corpus into two thirds for training (14,941,100) and one third for testing (7,470,501). Most records in PubMed are manually assigned one or more MeSH headings. There are over 27 thousand terms in the MeSH vocabulary, giving rise to as many possible classification problems on this corpus. In a previous work[14], we studied 20 MeSH terms representing a range of frequencies. Each of these MeSH terms is listed in Table 2 along with its frequency in the studied version of PubMed. Also shown in Table 2 are seven additional higher frequency MeSH terms that were selected for algorithm timing.

Table 2. MeSH terms used in this study of SGD-SVM on the PubMed corpus. Given for each MeSH term is the number of records indexed with that MeSH term and the percent this is of the total.

Set	MeSH Terms	Freq	Percent
M1	rats, wistar	180,179	0.80%
M2	myocardial infarction	131,491	0.59%
M3	blood platelets	63,061	0.28%
M4	serotonin	60,164	0.27%
M5	state medicine	43,787	0.20%
M6	urinary bladder	39,795	0.18%
M7	drosophila melanogaster	31,502	0.14%
M8	tryptophan	25,803	0.12%
M9	laparotomy	14,554	0.06%
M10	crowns	12,862	0.06%
M11	streptococcus mutans	6,623	0.03%
M12	infectious mononucleosis	6,593	0.03%
M13	mentors	6,425	0.03%
M14	blood banks	5,753	0.03%
M15	humeral fractures	5,447	0.02%
M16	tuberculosis, lymphnode	4,471	0.02%
M17	tooth discoloration	2,570	0.01%
M18	pentazocine	2,128	0.01%
M19	hepatitis e	1,845	0.01%
M20	genes, p16	1,782	0.01%
M21	pathological condition, signs and symptoms	3,819,888	17.04%
M22	therapeutics	2,898,880	12.93%
M23	pharmacologic actions	2,404,751	10.73%
M24	enzymes	2,167,158	9.67%
M25	population characteristics	1,242,430	5.54 %
M26	reproductive physiological phenomena	1,019,313	4.55%
M27	terpenes	217,959	0.97%

Results

SGD-SVM Experiments Three folds were used in all cross-validation experiments so that we set $J = 3$ and, where used, $K = 3$. Cross validation was used for all databases except WebKB and 20 Newsgroups which are already partitioned into training and test sets. In these latter two cases we still did the learning with an optimal number of iterations based on holding out a third of the training set to determine it versus doing the learning based on a fixed number of nine iterations over the training data.

Table 3. Results on small corpora for SGD-SVM with cross-validation and SGD-SVM with a fixed 9 iterations. Each row in the table contains the averages over all classification problems defined in that corpus. The last row averages over the corpora.

Set	SVM-SGD Cross Validation			SVM-SGD with Fixed 9 iterations	
	AP	BE	$Iter_{opt}$	AP	BE
Rebase	0.84	0.80	5.3	0.85	0.80
MDR	0.95	0.92	20	0.94	0.91
20News	0.84	0.80	12.2	0.85	0.81
Reuters	0.93	0.89	14	0.93	0.90
WebKB	0.77	0.73	3.8	0.77	0.72
Industry	0.86	0.82	9.7	0.92	0.88
Ave	0.87	0.83	10.8	0.88	0.84

Table 4. Results on the PubMed corpus for SGD-SVM with cross-validation and SGD-SVM with a fixed 8 iterations. The last row averages over the twenty MeSH terms.

MeSH Set	SVM SGD Cross Validation			SVM SGD with 8 iterations	
	AP	BE	$Iter_{opt}$	AP	BE
M1	0.502	0.510	11	0.495	0.508
M2	0.714	0.716	5	0.720	0.715
M3	0.662	0.691	5	0.659	0.690
M4	0.670	0.682	6	0.666	0.682
M5	0.284	0.358	10	0.284	0.356
M6	0.549	0.584	6	0.544	0.585
M7	0.667	0.659	8	0.667	0.659
M8	0.603	0.612	9	0.602	0.613
M9	0.253	0.330	13	0.250	0.332
M10	0.593	0.603	18	0.591	0.597
M11	0.794	0.805	4	0.787	0.799
M12	0.711	0.734	3	0.701	0.724
M13	0.410	0.477	8	0.410	0.477
M14	0.372	0.432	5	0.381	0.433
M15	0.585	0.613	11	0.592	0.620
M16	0.462	0.496	9	0.464	0.499
M17	0.465	0.508	12	0.469	0.516
M18	0.702	0.718	13	0.701	0.714
M19	0.737	0.772	15	0.735	0.784
M20	0.316	0.421	3	0.304	0.414
Ave	0.552	0.586	8.7	0.551	0.586

The results in Table 3 and Table 4 strongly support our contention that a fixed number of iterations gives as good performance for SGD-SVM as using held out data to determine an optimal number of iterations. We also compared SGD-SVM with three published SVM algorithms designed to work well on large training sets. These algorithms are PROBE[30], SVMPerf[31] and LibLinear[32]. Results for all algorithms are in Table 5. No single algorithm is superior to the others on all MeSH terms and SGD-SVM with a fixed 8 iterations is competitive with the others.

Table 5. Results on the PubMed corpus for the four algorithms: SGD-SVM with a fixed 8 iterations, PROBE, SVMperf and LibLinear. For each of the MeSH terms the MAP and BE are reported for a three-fold cross validation and the same folds were used for all the algorithms. The last row averages over the twenty MeSH terms.

	SGD-SVM		PROBE		SVMPerf		LibLinear	
	MAP	BE	MAP	BE	MAP	BE	MAP	BE
M1	0.495	0.508	0.482	0.495	0.472	0.495	0.512	0.507
M2	0.720	0.715	0.723	0.711	0.707	0.697	0.753	0.711
M3	0.659	0.690	0.671	0.689	0.648	0.673	0.693	0.683
M4	0.666	0.682	0.680	0.684	0.651	0.664	0.688	0.671
M5	0.284	0.356	0.308	0.375	0.285	0.350	0.291	0.338
M6	0.544	0.585	0.536	0.570	0.521	0.562	0.553	0.548
M7	0.667	0.659	0.665	0.645	0.633	0.635	0.693	0.642
M8	0.602	0.613	0.591	0.609	0.575	0.599	0.606	0.594
M9	0.250	0.332	0.276	0.340	0.255	0.326	0.308	0.355
M10	0.591	0.597	0.594	0.599	0.583	0.603	0.582	0.569
M11	0.787	0.799	0.777	0.790	0.751	0.773	0.791	0.791
M12	0.701	0.724	0.706	0.729	0.666	0.706	0.725	0.727
M13	0.410	0.477	0.418	0.484	0.389	0.451	0.425	0.486
M14	0.381	0.433	0.381	0.437	0.339	0.424	0.378	0.409
M15	0.592	0.620	0.577	0.606	0.557	0.590	0.568	0.586
M16	0.464	0.499	0.479	0.499	0.428	0.467	0.461	0.495
M17	0.469	0.516	0.443	0.502	0.399	0.463	0.400	0.452
M18	0.701	0.714	0.683	0.690	0.571	0.570	0.701	0.687
M19	0.735	0.784	0.714	0.764	0.717	0.772	0.716	0.737
M20	0.304	0.414	0.311	0.401	0.319	0.401	0.374	0.433
Ave	0.551	0.586	0.551	0.581	0.523	0.561	0.561	0.571

In doing the computations for Table 5 it became evident that SGD-SVM was much faster than the other algorithms. To investigate the issue of computation time further we chose seven additional MeSH terms with high frequencies (thus large positive sets) where the calculation is more difficult. Statistics on these MeSH terms are listed in Table 2. Both performance and elapsed time (wall clock) are shown in Table 6. Two things stand out in Table 6. First, SGD-SVM performs better than PROBE or SVMPerf on these problems and at least as good as LibLinear. Second, PROBE averages between 4 and 5 hours, LibLinear between 8 and 9 hours and SVMPerf may take much longer (we stopped any calculation over 30 hours) for these problems. This in contrast with SGD-SVM which takes 10 minutes or less. We conclude from these experiments that in all circumstances, particularly for large training sets with positive sets of significantly large size, the SGD-SVM algorithm with a fixed 8 iterations is the preferred method.

Table 6. BE performance and elapsed time for classification of high frequency MeSH terms. Here SGD-SVM is with a fixed 8 iterations. All timings were conducted using a standalone multi-core computer with Intel Xeon CPUs with a clock speed of 2.67 GHz and 48 gigabytes of RAM. NA=not available.

	SVM-SGD		PROBE		SVMPerF		LibLinear	
	BE	Time	BE	TIME	BE	TIME	BE	TIME
M21	0.637	9 Min	0.592	284 Min	NA	>30 Hr	0.636	564 Min
M22	0.616	10 Min	0.584	283 Min	NA	>30 Hr	0.619	553 Min
M23	0.616	10 Min	0.574	291 Min	NA	>30 Hr	0.614	547 Min
M24	0.768	9 Min	0.722	256 Min	NA	>30 Hr	0.762	557 Min
M25	0.507	9 Min	0.499	250 Min	NA	>30 Hr	0.521	535 Min
M26	0.709	9 Min	0.690	290 Min	0.689	737 Min	0.702	534 Min
M27	0.714	9 Min	0.671	308 Min	0.698	191 Min	0.689	516 Min
Ave	0.652	9.3 Min	0.619	208 Min	NA	NA	0.649	543 Min

Application to MeSH Assignment Given the speed and accuracy of SGD-SVM, we were led to contemplate calculations which would have been unthinkable in the past. There are over 27 thousand MeSH terms, but it is not difficult to train classifiers for all of them. The question is whether such a set of classifiers will yield superior performance in predicting MeSH assignments for unseen documents. To answer this question we decided to perform an experiment. There are two sets of PubMed documents on which experiments have been done in the past and with which we can compare our results. One set is known as NLM2007, is available at[33] and consists of 200 PubMed records originally selected in 1999 and used as a bench mark by several studies[2, 34, 35]. The second set is L1000, a set of 1,000 randomly selected PubMed records, created and studied in[34] and available at[36].

Based on these considerations we held out the two test sets and used the remainder of the March 2013 MEDLINE as training data. For each MeSH term we randomly divided the training set into two disjoint sets with each half having, as nearly as possible, the same number of documents with the MeSH term assigned and without the MeSH term assigned. We then trained one SGD-SVM classifier on each half. Once a classifier was trained we applied it to produce scores on the half of the documents where it had not been trained. Finally, we applied the Pool Adjacent Violators (PAV) algorithm to these scores to convert them to probabilities. This algorithm produces a probability function of scores that is non-decreasing as a function of score and gives a best fit to the actual data. Best fit here means if the probabilities are looked at as predictions of whether the MeSH term is assigned to a document or not, then the actual assignments observed in the data have maximum probability. No other non-decreasing probability function of score could assign a higher probability to the data. Since we have two trainings, one on each half of the training data, we have two probability functions. We then score the test documents for each training and convert each into a probability based on its PAV function and average these two probabilities to obtain our probability estimate that the MeSH term will be assigned to the test document. Space does not allow more detail regarding PAV, but we refer the reader to[25].

Evaluation. We give four measures of performance: precision, recall, F_1-score, and mean average precision (MAP) which evaluate success in automatic MeSH term assignment to MEDLINE citations. We use these metrics because they have been used for the same task in previous studies. These metrics are standards in the field and are described in[27]. For a given test document precision at rank 25 is the fraction of MeSH terms in the top ranked 25 which were assigned to that document by the human indexers and recall at rank 25 is the fraction of MeSH terms assigned by human indexers to that document which appear in the top 25 ranks of the prediction. The F_1-score at rank 25 is the harmonic mean of the precision and recall at rank 25. The average precision of a ranking for a document is the average of all precisions computed at ranks containing a correctly assigned MeSH term for that document. Finally, for a given set of test documents we compute these four measures for each document and average the results for each measure over all documents in the set and report this average. For average precisions this final average has been given the name of mean average precision or MAP.

We compare our approach to five other methods which were previously published in[34]. The first one is NLM's MTI system[2]. The second method is known as reflective random indexing[35]. The third and fourth are k-nearest neighbor methods. For these methods the neighbors are computed using the algorithm reported in[37]. The value of k determined to be optimal for both frequency and similarity approaches was studied in[34] and found to be 20. The fifth method is based upon a learning-to-rank algorithm which begins with the features used for ranking in the k-nearest neighbor methods and adds a number of other features and learns a weighting for these features that allows an improved ranking of the MeSH terms assigned to the k-nearest neighbors for all training documents at once. For more details see[34]. Table 7 and Table 8 list the results for the databases NLM2007 and L1000, respectively.

Table 7. Precision, recall, F-score over top 25 predictions, and MAP for different methods on the set NLP2007.

	Prec	Recall	F-Score	MAP
MTI	0.318	0.574	0.409	0.450
Reflective random indexing	0.372	0.575	0.451	N/A
Neighborhood frequency	0.369	0.674	0.476	0.598
Neighborhood familiarly	0.376	0.677	0.483	0.604
Learning-to-rank algorithm	0.390	0.712	0.504	0.626
SGD-SVM	0.390	0.712	0.504	0.640
SGD-SVM with PAV	0.405	0.740	0.524	0.681

Table 8. Precision, recall, F1-score over top 25 predictions, and MAP for different methods on the L1000 set.

	Prec	Recall	F-Score	MAP
MTI	0.302	0.583	0.398	0.462
Neighborhood frequency	0.329	0.679	0.443	0.584
Neighborhood similarity	0.333	0.687	0.449	0.591
Learning-to-rank algorithm	0.347	0.714	0.469	0.615
SGD-SVM	0.346	0.712	0.465	0.638
SGD-SVM with PAV	0.368	0.756	0.495	0.676

In these tables all results compared with SGD-SVM are taken from[34]. Two things need to be noted regarding these comparisons. First, for both NLM2007 and L1000 we used the PubMed IDs to find the current forms of the documents and our results are based on the current indexing of these documents. This is the only reasonable thing to do as our training data is all based on the current indexing. This current indexing could conceivably involve some changes that could affect the difficulty in assigning MeSH terms to a document. Since we produced the k-nearest neighbor results used in[34] and quoted in these tables, we recomputed these numbers as a check on any such possible change. We found the four measures to be identical to the numbers given here for the neighborhood frequency method for both NM2007 and L1000. For the neighborhood similarity method we found small variations. For NLM2007 the current F1-score is 0.480 (down 0.003) and the current MAP is 0.605 (up 0.001) and for L1000 the current F1-score is 0.449 (unchanged) and the current MAP is 0.592 (up 0.001). We present these results as evidence that the indexing task for NLM2007 and L1000, as they appear in the 2013 data used for this study, is substantially what it was when studied in[34] and in any case has not gotten easier. The second point we wish to make is that previous studies were limited to rankings of MeSH terms coming from a short list of neighbor documents (generally the top 20) and this limited the number of participants in the ranking to compute the MAP values reported in the tables. In our approach all MeSH are ranked for each document and this allows for a slight improvement in the MAP numbers.

We applied statistical tests to compare the MAP values produced by SGD-SVM with PAV with the Learning-to-rank results. Each method gives a figure for the average precision of MeSH assignment to a document. These pairs over all the documents in a set are then tested. The test methods and the corresponding p-values are given in Table 9.

Table 9. Comparison of SGD-SVM with PAV with Learning-to-rank. We applied the sign test, the Wilcoxon signed rank test, and the paired t test.

	NLM 2007	L1000
Sign test	1.2×10^{-9}	1.4×10^{-39}
Wilcoxon	6.0×10^{-12}	7.9×10^{-59}
Paired t test	2.5×10^{-4}	$<10^{-4}$

Discussion

From a theoretical point of view it is interesting to ask why SGD with early stopping works. In this regard it is useful to point out that what Zhang[17] calls SGD with early stopping is called by Collobert and Bengio[38] the margin perceptron algorithm. The latter authors point out that the wider the margin the better the performance is likely to be and that the margin is preserved by a small value of λ and by a small number of iterations. The small number of iterations is achieved by early stopping. However, as far as we can discern, it has not previously been appreciated that early stopping can be achieved with a fixed number of iterations for a whole class of problems.

The practical consequence of SGD with early stopping at a fixed number of passes over the data is an approximate halving of the training time. The first run with held out data to determine the optimal number of passes over the data proves unnecessary as documented in Table 3 and Table 4. SGD with early stopping is already a very useful approach on problems with large training data and using fixed iterations only adds to the benefit. With this approach there is no need to reduce the size of the training set to make the calculations feasible as done in[39]. We used unigram and bigram features based on our own experience that this gives good performance[28]. We see that[39] also used unigram and bigram features, but due to our much larger training set sizes we had to deal with a much larger feature set (over 70 million features as compared with their approximately 2 million). In spite of this we can perform a single training run in 10 minutes and perform training over all MeSH terms in a couple of days on our compute farm using approximately 200 cpu's. While at this point we have not been able to directly compare our approach to that of[39], given that they use the same feature types and use a standard SVM approach, we would expect the two methods to perform at a similar level. The results of Table 5 and Table 6 support this conclusion.

We tested our approach on the test sets NLM2007 and L1000 because there are published results for several different methods on these sets[34] and because the particular method of testing on the top 25 predicted MeSH terms is very relevant to the MTI system at NLM. The suggestions to the MeSH indexers by the MTI system[6] are generally on the order of the top 25 terms. The precision, recall and F1 scores given in Table 7 and Table 8 are all computed for the top 25 predicted MeSH terms.

Conclusions

First, SGD with early stopping at a fixed number of iterations is an accurate and fast way to train a SVM classifier for large training sets. Second, when SGD-SVM is combined with the PAV algorithm the results on two previously studied test sets are superior to published results. We are currently collaborating with James Mork and Alan Aronson at the NLM testing whether results from SGD-SVM can be used to improve the MTI system.

Acknowledgement This research was supported by the Intramural Research Program of the NIH, National Library of Medicine.

References

1. Aronson AR, Bodenreider O, Chang HF, Humphrey SM, Mork JG, Nelson SJ, et al., The nlm indexing initiative. American Medical Informatics 2000 Annual Symposium; 2000; Los Angeles, CA: American Medical Informatics Association.
2. Aronson AR, Mork JG, Gay CW, Humphrey SM, Rogers WJ. The nlm indexing initiative's medical text indexer. Stud Health Technol Inform. 2004; 107:268-72.
3. Herskovic J, Cohen T, Subramanian D, Iyengar M, Smith J, Bernstam E. Medrank: Using graph-based concept ranking to index biomedical texts. Int J Med Inform. 2011; 80(6):431-41.
4. Jimeno-Yepes A, Plaza L, Mork J, Aronson A, Díaz A. Mesh indexing based on automatically generated summaries. BMC Bioinformatics. 2013; 14(208).
5. Kim W, Aronson AR, Wilbur WJ, Automatic mesh term assignment and quality assessment. Proc AMIA Symp; 2001; Washington, D.C.
6. Mork JG, Jimeno Yepes AJ, Aronson AR. The nlm medical text indexer system for indexing biomedical literature. BioASQ; Valencia, Spain; 2013.
7. Neveol A, Shooshan S, Humphrey S, Rindflesh T, Aronson A, Multiple approaches to fine-grained indexing of the biomedical literature. Pacific Symposium on Biocomputing; 2007.
8. Névéol A, Shooshan S, Mork J, Aronson A, Fine-grained indexing of the biomedical literature: Mesh subheading attachment for a medline indexing tool. AMIA Annu Symp Proc; 2007.
9. Neveol A, Shooshan SE, Humphrey SM, Mork JG, Aronson AR. A recent advance in the automatic indexing of the biomedical literature. Journal of biomedical informatics. 2009; 42(5).
10. Jimeno-Yepes A, Mork J, Demner-Fushman D, Aronson A, Automatic algorithm selection for mesh heading indexing based on meta-learning. Fourth International Symposium on Languages in Biology and Medicine; 2011; Singapore.
11. Jimeno-Yepes A, Mork J, Demner-Fushman D, Aronson A. A one-size-fits-all indexing method does not exist: Automatic selection based on meta-learning. Journal of Computing Science and Engineering. 2012; 6(2):151-60.
12. Jimeno-Yepes A, Mork J, Demner-Fushman D, Aronson A, Comparison and combination of several mesh indexing approaches. AMIA Annual Symposium; 2013; Washington, D.C.: American Medical Informatics Association.
13. Jimeno-Yepes A, Mork J, Wilkowski B, Demner-Fushman D, Aronson A, Medline mesh indexing: Lessons learned from machine learning and future directions. ACM International Health Informatics (IHI) Symposium; 2012.

14. Sohn S, Kim W, Comeau DC, Wilbur WJ. Optimal training sets for bayesian prediction of mesh assignment. J Am Med Inform Assoc. 2008 Jul-Aug; 15(4):546-53.
15. Trieschnigg D, Pezik P, Lee V, de Jong F, Kraaij W, Rebholz-Schuhmann D. Mesh up: Effective mesh text classification for improved document retrieval. Bioinformatics. 2009; 25(11):1412-8.
16. Wikipedia_SGD. Stochastic gradient descent. 2013; Available from: http://en.wikipedia.org/wiki/Stochastic_gradient_descent#cite_ref-6.
17. Zhang T, Solving large scale linear prediction problems using stochastic gradient descent algorithms. Twenty-first International Conference on Machine learning; 2004: Omnipress.
18. Finkel JR, Kleeman A, Manning CD, Efficient, feature-based, conditional random field parsing. ACL-08: HLT; 2008; Columbus, Ohio.
19. Tsuruoka Y, Tsujii Ji, Ananiadou S, Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP; 2009; Suntec, Singapore: ACL.
20. Bandos TV, Camps-Valls G, Soria-Olivas E. Letters: Statistical criteria for early-stopping of support vector machines. Neurocomput. 2007; 70(13-15):2588-92.
21. Prechelt L. Automatic early stopping using cross validation: Quantifying the criteria. Neural Networks. 1998; 11:761-7.
22. Raskutti G, Wainwright MJ, Yu B. Early stopping and non-parametric regression: An optimal data-dependent stopping rule. J Mach Learn Res. 2014; 15(1):335-66.
23. Ayer M, Brunk HD, Ewing GM, Reid WT, Silverman E. An empirical distribution function for sampling with incomplete information. Annals of Mathematical Statistics. 1954; 26:641-7.
24. Barlow RE, Bartholomew DJ, Bremner JM, Brunk HD. Statistical inference under order restrictions. The theory and application of isotonic regression. New York: John Wiley & Sons; 1972.
25. Wilbur WJ, Yeganova L, Kim W. The synergy between pav and adaboost. Machine Learning. 2005; 61:71-103.
26. Zadronzny B, Elkan C. Transforming classifier scores into accurate multiclass probability estimates. Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; Edmonton, Alberta, Canada: ACM; 2002. p. 694-9.
27. Manning CD, Raghavan P, Schütze H. Introduction to information retrieval. Cambridge, England: Cambridge University Press; 2009.
28. Wilbur WJ, Kim W. The ineffectiveness of within-document term frequency in text classification. Information Retrieval. 2009; 12:509-25.
29. PubMed H. Pubmed help [internet]. Bethesda, MD: National Center for Biotechnology Information (US); 2013. Available from: http://www.ncbi.nlm.nih.gov/books/NBK3830/.
30. Smith L, Kim W, Wilbur WJ. Probe: Periodic random orbiter algorithm for machine learning2012.
31. Joachims T. Training linear svms in linear time. Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining; Philadelphia, PA, USA: ACM; 2006.
32. Fan R-E, Chang K-W, Hsieh C-J, Wang X-R, Lin C-J. Liblinear: A library for large linear classification The Journal of Machine Learning Research. 2008; 9.
33. Aronson AR. 200 medline citations test collection Bethesda, MD: National Library of Medicine; 2013 [cited 2013]; Available from: http://ii.nlm.nih.gov/DataSets/index.shtml#200MEDLINE.
34. Huang M, Neveol A, Lu Z. Recommending mesh terms for annotating biomedical articles. J Am Med Inform Assoc. 2011 Sep-Oct; 18(5):660-7.
35. Vasuki V, Cohen T. Reflective random indexing for semi-automatic indexing of the biomedical literature. J Biomed Inform. 2010; 43(5):694-700.
36. Lu Z. Recommending mesh data sets. Bethesda, MD: National Library of Medicine; 2013; Available from: http://www.ncbi.nlm.nih.gov/CBBresearch/Lu/indexing/.
37. Lin J, Wilbur WJ. Pubmed related articles: A probabilistic topic-based model for content similarity. BMC Bioinformatics. 2007; 8:423.
38. Collobert R, Bengio S. Links between perceptrons, mlps and svms. 21st International Conference on Machine Learning; Banff, CA; 2004.
39. Tsoumakas G, Laliotis M, Markantonatos N, Vlahavas I. Large-scale semantic indexing of biomedical publications at bioasq. BioASQ 2014 Valencia, Spain; 2013.