Linear Discriminant Text Classification in High Dimension

András Kornai[1] and J. Michael Richards[2]

[1] Northern Light Technology
222 3rd Street, Cambridge MA 02142
[2] PPD Informatics/Belmont Research
84 Sherman St, Cambridge MA 02140

Abstract. Linear Discriminant (LD) techniques are typically used in pattern recognition tasks when there are many ($n >> 10^4$) datapoints in low-dimensional ($d < 10^2$) space. In this paper we argue on theoretical grounds that LD is in fact more appropriate when training data is sparse, and the dimension of the space is extremely high. To support this conclusion we present experimental results on a medical text classification problem of great practical importance, autocoding of adverse event reports. We trained and tested LD-based systems for a variety of classification schemes widely used in the clinical drug trial process (COSTART, WHOART, HARTS, and MedDRA) and obtained significant reduction in the rate of misclassification compared both to generic Bayesian machine-learning techniques and to the current generation of domain-specific autocoders based on string matching.

1 Introduction

Linear Discriminant (LD) techniques, introduced by Fisher [8,9], have been a standard technique in pattern classification [14] even before they received their modern formulation in Rao [20]. Today, virtually all speech and character recognition systems employ some form of LD analysis, though it is generally incorporated in the signal processing front-end and viewed simply as a data reduction step. LD may be more effective than Karhunen-Loèwe transform, the conclusion reached by [3], but overall it remains just one of the many feature extraction steps, contributing to classification only indirectly. Since the seventies, perhaps as a result of the strongly argued criticisms levelled against perceptrons by Minsky and Papert [18], the direct use of linear classification techniques has largely given way to more complex (e.g. polynomial) classifiers [19].

In Section 2 we introduce a problem domain, *autocoding,* and give an overview of the current generation of autocoders. The linear classification methods we apply to this task are described in Section 3, where we also discuss why the issues raised by Minsky and Papert are irrelevant for sparse data in high dimension. Our results, showing that linear classification is markedly superior, are presented in Section 4. In the concluding Section 5 we discuss why the perceptron algorithm, the granddaddy of all learning algorithms based on error minimization, performs so well.

2 Autocoding

In the clinical trial process for new drugs, health professionals routinely generate *adverse event reports* that describe side effects of the drug under trial. Coding these reports or *verbatims* to an established set of codepoints as provided by various standardized classification schemes such as *WHOART, COSTART, HARTS,* or *MedDRA* can be thought of as an instance of the general information retrieval/text classification problem, or more specifically, as an instance of the message routing (MR) problem [13]. In supervised MR we are faced with the following problem: given some categories $C = \{C_1, C_2, \ldots, C_k\}$ and some "truthed" documents $F = \{F_1, F_2, \ldots, F_n\}$ with their true categories $t(F_i) \subset C$, for any new document F_{n+1}, find the most likely value of $t(F_{n+1})$, i.e. the category or categories associated to F_{n+1}. In a typical MR system files are first reduced to multisets or *bags* of words or word stems in the preprocessing stage, so that the task becomes classifying these bags of words, or stems, rather than the original files. Given an arbitrary fixed ordering of words/stems, e.g. as created by a perfect hash function, bags are in one to one correspondence with vectors of a large dimensionality d (the number of words, usually several thousand to a few hundred thousand) having only nonnegative integer coefficients called in these applications the *counts.* Looked at this way, MR is a classification problem in Euclidean space, and dimension reduction, usually in the form of Singular Value Decomposition (SVD) is a key aspect of successful classification [5,11].

Currently, pharmaceutical companies and clinical research organizations largely rely on the expertise of human coders familiar with these coding schemes [7], employing MR techniques only very conservatively, primarily to assure that similar verbatims receive the same code. To check for similarity, the current generation of autocoders perform some combination of the following abstraction steps [12,10]

1. special character normalization
2. spell correction
3. acronym expansion
4. undoing abbreviations
5. capitalization
6. morphological analysis (stemming)
7. synonym-based data enrichment
8. word order normalization

in the hope that the result matches an already coded verbatim listed in the *synonym table* shipped with the system. These autocoders are therefore nearest neighbor classifiers, but the abstract string-similarity neighborhoods around the verbatims in the synonym table are very small. Since unseen strings will remain unclassified if they do not fall into any of these neighborhoods, the current generation of autocoders often reject (leave unclassified)

85% of the data. This is not to say that such techniques are useless (see [6] for a detailed study of their impact on queries in the biomedical domain), but they represent an extreme tradeoff between error and reject rates. In a machine intelligence context (and also for practical purposes) we are interested in techniques that can generalize better to data not seen in training, even at the cost of increased error. Our system contains only steps 1 (substituting "@" by `AT`, "#" by `NUMBER` etc.), 5, and 6 (a variant of the Porter stemming algorithm), since the other steps had insignificant impact on performance.

The two most salient properties that set autocoding apart from the bulk of the work on IR and message routing are that (1) the messages tend to be fairly short: in our samples, the average verbatim is less than 5 words and (2) the verbatims by no means conform to the standards of English syntax. In our work, a considerable portion, over 10%, of the verbatims is in German, French, or some other language, and the source language is not known in advance for the autocoder. Verbatims also tend to be heavily abbreviated, both in their syntax (`ALLERGY ARMS AND LEGS` instead of `SKIN ALLERGY ON THE ARMS AND LEGS`) and at the word level (`POSS. INTERSTITIAL INFLAM.` instead of `POSSIBLE INTERSTITIAL INFLAMMATION`), have nonstandard word order (`ACNE FACIAL RECURRENT` instead of `RECURRENT FACIAL ACNE`), and of course contain a large number of medical terms not found in standard dictionaries. These difficulties extend well beyond adverse event coding: similar issues need to be faced in coding to disease, symptom, or procedure classifications such as *DSM4, ICD10* or *ICPC*.

3 Methods

It has long been noted that word pairs and in general word n-grams are more informative than word unigrams alone. However, on a space encompassing n-grams the direct use of generalized matrix inversion techniques would have prohibitive storage requirements: for a typical vocabulary of 20k words we would need 400 megabytes for bigrams and 8 terabytes for trigrams. In addition to this *storage problem*, there is the more subtle but even more dangerous *data problem:* corpora for reliably estimating 8 trillion datapoints are simply not available. One way out of this problem is replacing measured values (frequency ratio estimates of probabilities) by computed values, using various backoff schemes [4]. This is a very active area of research, especially in language modeling for speech and character recognition, where corpora with $10^8 - 10^9$ words are now routinely available. Adverse event data are not available in these quantities: a typical clinical trial will generate only a few thousand words. To escape the data problem, we only use actual counts, "estimating" the probability of non-occurring n-grams by zero.

We solve the storage problem by a carefully designed sparse vector framework, which employs symbolic techniques to such an extent that we found it expedient to implement it in Java. While Java is currently not well suited to large-scale numerical work (see e.g. `math.nist.gov/javanumerics`), in the sparse vector library we incur only a constant factor overhead, so that moving from unigrams (words) to n-grams scales linearly with n. This is particularly important, since the dimension d of our sparse vectors is $\sum_{i=0}^{n} V^i \sim V^n$, so as n moves from 3 to 20, d moves from 10^{13} to 10^{86}. In this paper we will ignore the implementation details and simply assume that verbatims are characterized by *count vectors* that contain all i-gram counts for each verbatim for $1 \leq i \leq n$, and codepoints are characterized by *weight vectors* w_j that store the relevance $W_{R,j}$ of the i-gram R to the jth class. For each class, we can also employ a 0th element $w_{0,j} = -b_j$ called the *bias*. We use standard *linear machines* where the result of the classification is defined as the j for which the scalar product (w_j, c) with the count vector c is maximal (after subtracting the biases).

Since Minsky and Papert [18] the central objection to linear classifiers has been the observation that in normal d-dimensional space the sets of points corresponding to different classes tend not to be linearly separable. Indeed, data sets obtained by measuring physical parameters of objects, such as the Fisher iris data, are almost always “cloudy”, partly because of measurement errors, and partly because of the inherent variability of the real-world process that created the objects. Also, such data tend to have low dimension, since measuring different dimensions typically requires separate instrumentation. In our case, however, d is 10^{13} or even higher, and only artificially constructed, as opposed to actually observed, data sets will manifest nonseparable behavior. To see what is involved here, consider two classes C and D with count vectors c_i and d_j. For these classes to be nonseparable, the convex hulls of the c_i and d_j need to intersect, meaning that we find two convex linear combinations $\sum \gamma_i c_i = \sum \delta_j d_j$. Taking the coordinates at an arbitrary n-gram R, the counts need to show the same equality, meaning that for each R some C counts need to bracket some D counts or conversely, some D counts need to bracket some C counts. There are many Rs where this routinely happens, in particular, counts for stopword unigrams are not expected to be separable. But overall nonseparability would require *all* counts to be nonseparable, which is contrary to experience. What we find in practice is that each class has its characteristic words and n-grams, most of which do not even appear in competing classes. The overall effect is that each class is concentrated in a low-dimensional ($d = 10^1 - 10^3$) subspace, and these subspaces are largely disjoint.

3.1 Shared components

The algorithm has two modes of operation: *training* and *classification*. In training mode, the input consists of verbatims and associated truth values, typically extracted from a database in a preparatory step, and the output is a file of weight vectors. In classification mode, the weight vectors produced during training are read in as input, and for each input verbatim a list of truth values and confidences is output. Most of the code, including the normalization steps described above, are shared between the trainer and the classifier. Major steps in training are as follows.

3.2 Preprocessing

We employ shallow parsing techniques [1] to identify chunks separated by hyphens, parentheses, and other punctuation. A novel aspect of this step is the *ranking* of the chunks according to their hypothesized hierarchical importance. Chunks shorter than four words that span the whole verbatim are considered *definitive* in the sense that they contain all the information necessary for classification and receive rank 0, partial chunks have rank 1 and higher. As Table 1 shows, the number of less valuable (higher ranked) chunks emerging from our shallow parser falls of rapidly with rank. The quality of the information contained in lower-ranking chunks is also rapidly decreasing.

	WHOART tokens	*WHOART* types	*COSTART* tokens	*COSTART* types
trainset	64765	64515	50301	8038
testset	7195	7195	5589	1718
rank 0	62636	62366	50264	7946
rank 1	10759	8630	1796	556
rank 2	649	524	92	38
rank 3	22	22	2	2
rank 4	2	2	0	0

Table 1. Distribution of chunk ranks

In one set of experiments, we obtained 93.1% correct classification using rank 0 chunks, but only 55.9% using rank 1 chunks *(COSTART)*, and 72.6% using rank 0 vs. 39.5% using rank 1 *(WHOART)*. It is therefore tempting to discard every chunk with rank ≥ 1. However, classifiers combining rank 0 and higher rank information with the proper weighting scheme in fact reduce the error rate by about 20%. This is less than the 40-56% reduction one may expect based on rank 1 alone, but of course these are not fully independent information sources, since the chunks at different ranks are highly correlated. Differential use of the chunks, using lower weights for higher ranks, is important not only in extracting the information, but also in protecting the weights based on rank 0 from contamination by higher rank material.

3.3 Initialization

We initialize the weight vectors based on the definitive rank 0 and rank 1 n-grams. This is, in essence, an "example-based learning" technique [2]. It is particularly important in our case because this step ensures that examples occurring only once get used in classification, and therefore our perceptron subsumes, as a special case, the traditional synonym table-driven methods.

Based on the seeds accumulated thus far, we perform statistical n-gram analysis, and add every i-gram that is *valuable* in the sense that it appears as part of a more than one definitive n-gram. The importance of this step is to enable generalization: without it, a verbatim that is missing from the training set can be found only if it has a sufficiently short and high-ranked substring in the training set. With the addition of valuable n-grams success rates on unseen verbatims go from 42.8% to 72.6% *(COSTART)*. The weight of the valuable i-grams, together with the original definitive n-grams, is clipped at 2, so as not to attribute undue importance to frequent examples.

The main novelty of our n-gram frequency analysis is that the sparse vector library enables us to implement it for n as high as we wish. Because the average verbatim is slightly less than five words long, we found no advantage in using $n > 4$, but in other message routing applications, such as the analysis of discharge reports, the ability to increase n is likely to provide additional benefit.

Our procedure is for each category to take the list of n-grams that appear $k > 1$ times, and assign a weight of k/F to each, where F is the count of the most frequent such n-grams. The example-based, valuable, and frequent lists are all filtered for stopwords. *Flat initialization* (without filtering stopwords and words that appear only once, and omitting the steps described above) would be a considerably simpler procedure, but it leads to noticeably worse iteration results: for example, in the `VISION` bodysystem (*WHOART*, 1461 training examples for 51 categories) Widrow-Hoff iteration yields 62.5% correct classification from flat initialization, and 67.5% from the more complex initialization described above.

3.4 Iteration

In machine learning systems the iterative step is usually performed via generalized inverse computation. Here the lack of concern for nonseparability enables us to implement this step more directly. We experimented with a variety of settings and update formulas, ranging from classic Widrow-Hoff perceptron to Winnow, but we report results only on Widrow-Hoff, since the choice between different update formulas had an order of magnitude less impact than the size effects discussed in Section 4. Iteration cuts error rates by a quarter *(COSTART)* to a third *(WHOART)*. In the course of experimenting with a variety of stopping criteria, we generally found that within the first k iterations (usually between 5 and 20) the one that was best on the training set was either the best one on the test set as well, or that the difference between the two was insignificant. Typically, very little improvement is seen after the first 3-7 iterations, suggesting that the system memorizes quite effectively what little training data is has.

An important negative finding concerning both initialization and iteration was that using non-zero biases (constant terms) does not improve the classification rate. Since in general a system with a larger number of parameters is expected to provide better results, the fact that bias-free linear machines offer as good classification as those using biases is strong evidence in favor of the geometrical picture offered at the beginning of this Section, namely that the different classes are comfortably segregated in different subspaces of the overall space.

4 Results

As in any classification task, the results of autocoding are heavily impacted by *size effects:* the less training data are available, and the more detailed the classification scheme, the higher the misclassification rate. For synonym-table based systems the results of autocoding also depend very heavily on the size of the synonym table and on the amount of repetition in the data, which can vary extensively: for example our raw *COSTART* data has over 56k verbatim tokens, but less than 10k different types, while our raw *WHOART* data has only 250 repeated tokens for over 72k types. The normal procedure is for each data set to randomly select 90% for training and 10% for testing: this would results in a test set that has over 65% overlap with the training set for *COSTART* but less than .5% overlap for *WHOART*. Thus, a table-driven system, which simply memorizes the training data, would have over 65% success on the raw *COSTART* data but less than 1% on *WHOART*.

To remove the effect of repetition, we `uniq`ed both training and test data, and omitted from the test set every verbatim that was present in the training set. This procedure of course decreases the absolute numbers we report, but the results provide a pure measure of generalization ability. To control for the widely different number of categories across different coding systems, we first grouped together codepoints belonging to the same *bodysystem,* which reduces the number of *COSTART* classification categories to 12, and that of *WHOART* categories to 32.

coding system	total verbatims	# of classes	test set size	% correct linear	%correct AV Discovery	% correct Bayesian
WHOART	7794	32	865	84.9	58.5	66.3
COSTART	8038	12	1718	95.7	61.2	80.7

Table 2. System comparison with few categories

At this level of granularity, the linear classifier already cuts down by half *(WHOART)* to three quarters *(COSTART)* the error rates obtained with AltaVista Discovery and with a high quality state of the art commercial Bayesian classifier. As we increased the number of categories, the performance difference became even more marked, and we had increasing difficulties providing enough real memory for these systems (especially the Bayesian) to complete training.

Because the bodysystem-level classifier performs quite well, we can use it as a master, whose output is used to select one of 12 bodysystem-specific slave classifiers which route the verbatim to its detail category. A direct classifier, routing from the root node to one of 730 leaves without the use of intermediary nodes, routes 85.9% of the test data correctly. The cascade has a combined classification rate of 85.8% at a reject rate of 1.3%. Since this is not a significant loss in accuracy, we use the cascade scheme extensively in the *MedDRA* classification task, where the number of codepoints is so large that the considerably lower memory requirements and improved speed of the cascade method become relevant. We experimented on several data sets coded according to different systems. The main parameters of these sets are summarized in Table 3 below. Train and test sets are `uniq`ed, non-overlapping.

coding system	total verbatims	# of classes	test set size	% correct
WHOART	7794	470	865	71.1
COSTART	8038	722	1718	83.2
HARTS	45465	1257	4546	76.6
MedDRA	156307	3818	13240	73.9

Table 3. Classification results

These numbers are about 20% better than those produced by state of the art machine learning systems without domain-specific knowledge, and on *COSTART* our classification rate is quite comparable to that reported in [15], a system which relies on an extensive effort of using the knowledge of doctors to handcraft a detailed synonym table (we found no results reported on *WHOART, HARTS,* or *MedDRA* so far). The 16% error rate we obtained by generic machine learning techniques is within striking distance of the 12% initial error rates of medically trained human coders who are well versed in the classification scheme. Our own efforts to hand-classify the data show that laymen without such training have error rates of 20% or higher, so the perceptron can already be argued to have acquired half of the domain-specific knowledge required for adverse event coding.

For practical purposes, the system needs to go beyond providing a hard classification decision and indicate the level of *confidence* it has. In linear classifiers, the most natural confidence measure would be the size of the scalar product: the larger this number, the more confidence we should have that our classification is correct. We found that by considering not only the top value, but a linear combination of the top two values, we obtain a better measure of overall confidence. By using this linear combination, we are in effect penalizing decisions where the second best choice comes very close to the first – adding subsequent terms for third, fourth, etc. gave no significant improvement over this simple idea.

Using the confidence as a rejection threshold, we can tune our system from rejecting nothing, the default mode, to rejecting as much as 80% of the data. While this may seem extreme, such high rejection rates are actually quite normal for synonym table-driven MedDRA autocoders. As Figure 1 shows, at 80% reject our error rate is 2.7%, one tenth of our error rate at 0 reject.

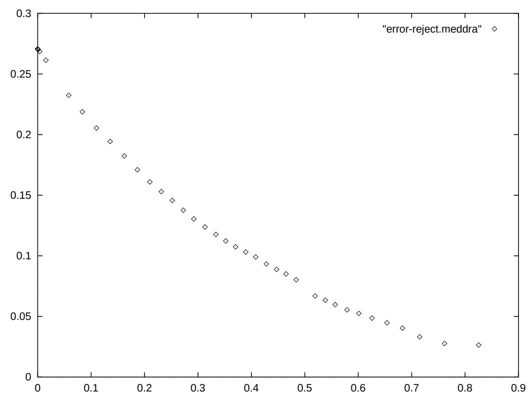

Fig. 1. Error rate as a function of rejection rate

5 Conclusions

The overall conclusion emerging from our experiments is that the remarkable power of simple perceptrons in this problem domain comes from a good match between the linear model and the actual structure of the data. When the dimension of the feature space is low and there are many datapoints, as in speech or character recognition, hyperplanes are simply incapable of modeling the actual complexity of the decision boundaries. In autocoding the dimension of the feature space is extremely high, and different classes are contained in largely disjoint subspaces, so that linear machines, with no bias, can already define decision boundaries which are as intricate as required. In effect, our perceptrons are nearest-neighbor classifiers, and the neighborhoods are well separated by coordinate indexes.

The details of the training, one-step vs. cascade, flat vs. complex initialization, Widrow-Hoff vs. Winnow, have only secondary effects. In the preprocessing stage, the use of standard string-normalization steps such as spell correction or undoing the abbreviations hardly had any impact. Whether deeper parsing would improve performance remains to be seen, but the non-standard syntax and style, not to speak of the multilingual nature, of adverse event reports create considerable difficulties for attacking the problem from this direction.

The main linguistic technique we found useful was morphological analysis (stemming), which yielded a small but consistent improvement of 4-5%. This improvement is not nearly as good as that found in [16] for the case of short queries and short documents, but is consistent with our observation that stemming and other string-normalization steps are weak in the sense that they leave most of the unseen data unclassified.

For TREC, [21] argued that "learning algorithms based on error minimization and numerical optimization are computationally intensive and prone to overfitting in a high dimensional feature space". For autocoding, both the computational feasibility of linear classification and its theoretical justification depend on the sparseness assumption. Given that the weight vectors have dimension as high as 10^{86}, if all the zero components were replaced by nonzero backoff numbers, computing a single scalar product could take longer than the lifetime of the universe. Linear separation can be guaranteed because any class will have distinctive n-grams, an assumption that would no longer hold if a large number of counts (or backoff values) would be nonzero. As for overfitting, the experimental results presented in Tables 2 and 3, with no overlap between test and train data, demonstrate quite clearly that this is not a valid concern here. While we have not run any experiments to specifically demonstrate this, we hypothesize that the overfitting observed in [21] is due to the fact that SVD actually makes the space less sparse than it is required for good linear separation.

The most salient property of autocoding data is that a single class (topic) rarely has more than a few hundred terms (n-grams) altogether, which obviates the need for feature selection, a crucial step in conceptually closely related TREC-based work [21,17]. However, the performance-limiting factor in MR is rarely the large classes with plenty of training data: the bulk of the error comes from those classes for which very little training was available. In this respect, conclusions drawn from large sparse datasets are even more relevant than conclusions based on a few well-represented categories.

Acknowledgment

This work was supported under SBIR grant # 2 R44 CA 65250 from the National Cancer Institute, National Institutes of Health. The authors would like to thank Michael Johnston, Jeremy Pool, and Lisa Stone for their help at various stages of the project.

References

1. Steven Abney. 1991. Parsing by chunks. In Robert Berwick, Steven Abney, and Carol Tenny, editors, *Principle-based parsing*. Kluwer Academic Publishers.

2. David W. Aha. 1991. Instance-based learning algorithms. *Machine Learning*, 61:37–66.
3. Peter F. Brown. 1987. *The acoustic-modelling problem in automatic speech recognition*. Ph.D. thesis, Carnegie-Mellon University.
4. Kenneth W. Church and William A. Gale. 1991. A comparison of the enhanced Good-Turing and deleted estimation methods for estimating probabilities of English bigrams. *Computer Speech and Language*, 5:19–54.
5. Christopher G. Chute and Yiming Yang. 1995. An overview of statistical methods for the classification and retrieval of patient events. *Methods of Information in Medicine*, 34:104–109.
6. Guy Divita, Allen C. Browne, and Thomas C. Rindflesch. 1998. Evaluating lexical variant generation to improve information retrieval. In *Proc. American Medical Informatics Association 1998 Annual Symposium*, Orlando, Florida.
7. Thérèse Dupin-Spriet and Alain Spriet. 1994. Coding errors: classification, detection, and prevention. *Drug Information Journal*, 28:787–790.
8. Ronald A. Fisher. 1936. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188.
9. Ronald A. Fisher. 1937. The statistical utilization of multiple measurements. *Annals of Eugenics*, 8:376–385.
10. Christian Fizames. 1997. How to improve the medical quality of the coding reports based on WHOART and COSTART use. *Drug Information Journal*, 31:85–92.
11. Stephen I. Gallant. 1995. Exemplar-based medical text classification. *Belmont Research SBIR Proposal 1 R43 CA 65250-01.*
12. Terry L. Gillum, Robert H. George, and Jack E. Leitmeyer. 1995. An autoencoder for clinical and regulatory data processing. *Drug Information Journal*, 29:107–113.
13. Donna K. Harman, editor. 1994. *The Second Text REtrieval Conference (TREC-2).* National Institute of Standards and Technology, Gaithersburg, Maryland.
14. Wilbur H. Highleyman. 1962. Linear decision functions with application to pattern recognition. *Proceedings of the IRE*, 50:1501–1514.
15. Michael. C. Joseph, Kathy Schoeffler, Peggy A. Doi, Helen Yefko, Cindy Engle, and Erika F. Nissman. 1991. An automated COSTART coding scheme. *Drug Information Journal*, 25:97–108.
16. Robert Krovetz. 1993. Viewing morphology as an inference process. In *Proceedings of SIGIR93*, pages 191–202.
17. David D. Lewis, Robert E. Schapire, James P. Callan, and Ron Papka. 1996. Training algorithms for linear text classifiers. In *Proceedings of SIGIR96*, pages 298–306.
18. Marvin Minsky and Seymour Papert. 1988. *Perceptons (2nd ed.).* MIT Press, Cambridge MA.
19. Jordan B. Pollack. 1989. No harm intended: A review of the perceptrons expanded edition. *Journal of Mathematical Psychology*, 33:358–365.
20. C. Radhakrishna Rao. 1965. *Linear statistical inference and its applications.* John Wiley, New York.
21. Hinrich Schütze, David A. Hull, and Jan O. Pedersen. 1995. A comparison of classifiers and document representations for the routing problem. In *Proceedings of SIGIR95*, pages 229–237.