Feature and Prototype Evolution for Nearest Neighbor Classification of Web Documents

Michelle Cheatham
Information Directorate
Air Force Research Laboratory
Wright-Patterson AFB, Ohio 45433
Email: michelle.cheatham@wpafb.af.mil

Mateen Rizki
Department of Computer Science
and Engineering
Wright State University
Dayton, Ohio 45432
Email: mateen.rizki@wright.edu

Abstract

A Nearest Neighbor Classifier (NNC) approaches the problem of text classification by computing a similarity metric between feature vector representations of an unknown document and a set of known prototype documents. The accuracy and speed of the NNC are dependent upon the choices of features and prototypes. In this work we consider the use of a genetic algorithm to optimize the feature and prototype sets for an NNC. We also examine whether simultaneously evolving the feature and prototype sets produces better results than sequential optimization.

1. Introduction

Google claims to have indexed over 8 billion documents, including PowerPoint, Word, text, pdf, and Flash as well as HTML files. With this amount of information available, it is becoming increasingly difficult to find specific data. As pointed out in [1], if web pages can be reliably categorized without human intervention, keyword searches and other techniques can then be applied to more limited categories instead of the entire web.

The problem of text categorization can be framed as a supervised learning task in which a classifier attempts to learn a relationship between a training set of documents and their categories. The classifier then applies that relationship to a test set of documents whose categories are not known. In this work, we have used a nearest neighbor classifier (NNC). NNCs are based on the appealingly simple notion that items which are close together in the feature space belong to the same class. In essence, a similarity measure is computed between an item whose class is unknown and a set of prototypes of known classification. The unknown item is assigned the classification of the most similar prototype. The NNC has been shown by many studies to perform as well or better than alternative classification schemes on both synthetic and real data sets ([2], [3]).

According to [4], the number of prototypes needed for an NNC to achieve a given accuracy increases exponentially with the number of irrelevant features. At the same time, both the computation time and memory required for the NNC increase with the number of prototypes and the size of the feature vector. In general, the p prototypes, each of length d (where d is the number of words in the dictionary), must be stored in memory. $O(pd)$ operations are required to compute the distances between the input and each prototype. Optimizing the feature and prototype sets of an NNC can therefore potentially improve both accuracy and efficiency.

There are many different approaches to optimizing the feature and prototype sets ([5], [6]). Of these, no one method dominates. Performance varies widely based on the nature of the data involved. We explore the utility of using a genetic algorithm (GA) to evolve the features and prototypes. A genetic algorithm is a heuristic search method that is an abstraction of Darwinian evolution. It uses reproduction, mutation, competition, and selection of a population of individuals to explore the search space. Our goal in this research is to answer two questions: Is a genetic algorithm an effective method for selecting the features and prototypes used by an NNC for text classification? and Does simultaneously evolving the feature and prototype sets produce better results than sequential optimization?

2. Related Work

Some research has been devoted to using a GA to select both the prototype and feature sets for an NNC. Kuncheva and Jain compare the performance of a GA for prototype and feature selection with established techniques such as using the condensed nearest neighbor rule and Wilson's method for prototype selection and sequential forward selection (SFS) for feature selection [3]. An individual in the population consisted of a binary string of length $p+f$, where p is the number of potential prototypes and f is the number of potential features. Their results indicate that the GA is the best choice among the methods studied when both accuracy and the number of features and prototypes are considered; however, SFS and Wilson's method used in conjunction produced a comparable performance. Other work has built upon this approach by attempting to add "intelligence" to the GA [7]. This approach uses orthogonal experimental design (OED) within the crossover operator to determine how much each gene is contributing to the overall fitness and chooses the most useful genes for propagation to the offspring. In this way, the algorithm attempts to explore the (very large) search space more efficiently than a tradition GA. The authors indicate that this method does indeed produce better results when accuracy and size are considered.

The method proposed here for simultaneously evolving the features and prototypes differs from the above methods in two ways. In the work above a single GA was used, with a single chromosome containing genes specifying both features and prototypes. In our approach, two nested evolutionary processes are used. A prototype set is evolved for each dictionary in the population. This allows a greater interrelation between dictionary and prototype set at the expense of increased computational complexity. In addition, the above research was done in domains that contain 20 or 30 potential features, while text classification is a more complex problem that typically involves hundreds or thousands of features.

3. Implementation

3.1. Preprocessing

The preprocessing phase consists of converting the documents from HTML to text and generating the feature set that will be optimized by the GA. We used a freeware tool called HTMLess, which can be downloaded at http://www.oz.net/sorth/media/htmls3d32.exe to do the conversion. The documents were then converted from free form text to feature vectors to be classified by the NNC. We have chosen binary feature vectors for our document representation, where a 1 indicates that the document contains a given word and a 0 that it does not. Other research done using this dataset has found boolean feature vectors to be amenable to classification [1].

There are two primary approaches to extracting a feature set for a given group of documents: using either the term frequency or the distribution of a term over the documents. The approach used in this system is to consider an averaged document frequency (DF). Specifically, the term frequency for each word in each document in the training set is computed and normalized with respect to the length of the document. Then the average of the normalized term frequency for each word over all documents is calculated. This has the effect of considering how important a word is in a particular document and how many documents contain that word, which are both indicative of the term's classification value. The words with the top average DF values were chosen as the initial features. In order to keep very common words from being included, a stopwords file consisting of the 1000 most commonly used words in the English language was used as a filter.

3.2. Genetic Algorithm

Notice the nested loops in Figure 1 that implement two cooperative genetic algorithms. The outer loop works to evolve sets of features while the inner loop works to co-evolve sets of prototypes. Both GA loops use random initialization, two-point crossover of random parents, a small mutation rate where the best individual is not subject to mutation, and elite selection.

The evaluation step requires a more in-depth explanation. The evaluation function tests each dictionary/prototype set combination by using it in an NNC to categorize a training set of documents. The fitness function used is given by Equation 1, where ν is the accuracy, F_n and P_n are the number of features and prototypes used, and F_m and P_m are the maximum number of features and prototypes available. The parameter α controls the emphasis placed on accuracy versus size, and β determines which is more important to optimize—the number of words in the dictionary or the number of prototypes. Evaluation is by far the most expensive step in the GA.

$$\alpha\nu + (1-\alpha)\left(\beta\left(-\tfrac{F_n}{F_m}\right) + (1-\beta)\left(-\tfrac{P_n}{P_m}\right)\right) \qquad (1)$$

3.3. System Complexity

Both the dictionaries and prototype sets are represented as bit strings, where a 1 indicates that the corresponding item is included in this individual and a 0 that it is not. This means that the search space for the dictionary is 2^w, where w is the size of the full dictionary. Similarly, the search space for the prototype sets contains 2^p items where p is the total number of potential prototypes pulled from all of the classes being categorized. The size of the combined search space is therefore 2^{w+p}. This is considerably larger than the $2^w + 2^p$ that would be the size if the features and prototypes were evolved sequentially.

Assuming the size of the dictionary and prototype populations are equal, and that both are evolved for the same number of generations, the complexity of evolving the

```
initialize the dictionary population with a random
distribution of ones and zeros

for (1:dictionary_epochs)
   double the dictionary population by randomly
   selecting parents and recombining using
   two-point crossover

   subject each gene in all but the best-performing
   dictionary to mutation with probability
   1/size(dictionary)

   assign a fitness score based on size and accuracy
   to each dictionary in conjunction with its best
   prototype set by executing the following {

      for (1:dictionary_population_size)
         initialize the prototype population with a
         random distribution of ones and zeros

         for (1:prototype_epochs)
            double the prototype population by randomly
            selecting parents and recombining using
            two-point crossover

            subject each gene in all but the best-
            performing prototype set to mutation
            with probability 1/number(prototypes)

            use the NNC with the current dictionary and
            prototype set to classify the training set
            and compute fitness based on accuracy

            reduce the prototype population by half by
            selecting the best-performing prototype sets
         end

         return the best-performing prototype set and
         its fitness score up to the dictionary->evaluate
         method
      end
   }

   reduce the dictionary population by half by selecting
   the best-performing dictionaries
 end
```

Fig. 1. Algorithm for Simultaneous Evolution of Features and Prototypes

feature and prototype sets simultaneously is of $O(s^2g^2)$ where s is the population size and g is the number of generations. Clearly, evolving the features and prototypes sequentially would be preferable. The complexity is then $O(2sg)$. It is possible, however, that the features and prototypes are so intricately interdependent that they must be evolved simultaneously. This is one of the questions that we hope to answer.

4. Experimental Setup

The BankSearch web page dataset was chosen as the document corpus. It is freely available online at http://www.pedal.reading.ac.uk/banksearchdataset. There are ten classes of documents, with 1000 documents per class (Table I). These documents are web pages that have been human-categorized as part of the Open Directory Project and Yahoo! Categories. This dataset supports classification tasks of varying levels of complexity. We tested our system on two groups of dissimilar documents (Commercial Banking and Soccer), two groups of similar documents (Building Societies and Insurance Agencies), and all ten groups.

TABLE I. Coarse and fine-grained classes within the BankSearch web page dataset.

Class	Specific Topic	General Topic
1	Commercial Bank	Banking and Finance
2	Building Societies	Banking and Finance
3	Insurance Agencies	Banking and Finance
4	Java	Programming Languages
5	C / C++	Programming Languages
6	Visual Basic	Programming Languages
7	Astronomy	Science
8	Biology	Science
9	Soccer	Sport
10	Motor Sport	Sport

The initial prototype set consists of 50 documents from each class. Another 50 documents from each group are used to generate the dictionary, which contains 200 potential features when two groups are being classified and 500 when all ten groups are being considered. The test set is comprised of a disjoint set of 100 documents from each class. The fitness function parameters α and β are both set to 0.5. A population size of 10 is used, and the number of generations is set to 30 except when both the feature and prototype sets were evolved. In that case, the number of generations was reduced from 30 to 15 due to the computation time required, particularly for the nested approach. Our results from evolving the features and prototypes alone indicate that the GA is able to produce good results within this time period however.

In order to effectively judge the contribution made by the GA, the accuracy of the NNC without any refinement of the dictionary and prototype set was first determined as a baseline for comparison. Then the prototype set was held fixed and the dictionary was evolved using the GA. Next the dictionary was fixed and the prototype set was evolved. It has been suggested by [8] that random search can sometimes perform comparably to a GA in feature selection. To determine if the genetic operators were helpful in generating good solutions, the crossover, mutation, and selection operators were removed and the population was randomly reinitialized every generation. The only memory between generations was the retention of the best-performing individual. This random method was used to generate both feature and prototype sets. Finally both features and prototypes were evolved. In the interest of time, we did not attempt to categorize all ten groups for this last experiment. Three different configurations were tested. In the first, the dictionary was evolved using the entire prototype set. Then the best dictionary was chosen and held constant while the prototype set was evolved.

TABLE II. Summary of Results

Test	Classes	Accuracy	Prototypes	Features
Unoptimized NNC	1 and 9	87	100	200
	2 and 3	81	100	200
	All	55	500	500
Dictionary Evolution	1 and 9	93	100	81
	2 and 3	81	100	81
	All	59	500	236
Randomized Dictionary	1 and 9	90	100	87
	2 and 3	82	100	88
Prototype Evolution	1 and 9	95	45	200
	2 and 3	88	49	200
	All	57	243	500
Randomized Prototypes	1 and 9	90	51	200
	2 and 3	80	52	200
Dictionary then Prototypes	1 and 9	94	52	82
	2 and 3	87	53	88
Prototypes then Dictionary	1 and 9	93	48	84
	2 and 3	85	50	81
Simultaneous Evolution	1 and 9	94	46	86
	2 and 3	86	49	82

The system was then modified to use the initial dictionary to evolve the prototype set first, and then use the chosen prototypes to optimize the dictionary. Finally, the system was set up to evolve the dictionary in an outer evolutionary process. For each generation in the dictionary evolution, a prototype set is evolved for each member of the dictionary population using another GA.

5. Results

Table II summarizes the results of the experiments. The values in the table are averaged over three trials. The accuracy is out of 100, the prototypes are out of 100 for the two-class case and 500 for all ten classes, and the features are out of 200 for the two-class case and 500 for all ten classes.

5.1. Unoptimized NNC

On average, the NNC classified the diverse documents with 87% accuracy and the similar documents with 81% accuracy. The accuracy on the full category set averaged 55%. It is somewhat surprising that NNC performed as well as it did in these tests. This is an indication that the strategy used to create the initial dictionary is at least marginally effective.

5.2. Feature Evolution

The results of this experiment are shown in the second and third rows of Table II. In all cases, the size of the feature set was reduced significantly (60% for the two-class cases and 47% for all ten classes). Accuracy improved by 7% over the unoptimized NNC for classes 1 and 9 and all ten classes, but remained the same for classes 2 and 3. The system was forced to make more accuracy/size tradeoffs when categorizing the similar documents. Altering the α parameter of the fitness function was unable to mitigate this problem. The accuracy of the randomly generated feature sets were comparable to the GA. However, the feature sets produced were larger on average. The fitness function of the GA contains a size penalty that allows it to focus the search on solutions that contain fewer redundant features.

TABLE III. Evolved Dictionary - Classes 1 and 9

Word	Class 1	Class 9	Word	Class 1	Class 9
100	25	12	**fans**	0	19
acorns	0	0	following	23	15
advice	19	1	goals	1	19
brazil	2	14	**investments**	26	0
clubs	2	28	limited	23	6
credit	48	1	**match**	1	33
dinamo	0	11	**news**	16	40
wednesday	0	11	**stadium**	0	16
payment	31	1	**property**	16	0
rights	8	20	subs	0	3
were	12	18	**savings**	47	1
security	43	0	uefa	0	26
germany	8	28	**repayment**	19	0

Table III is a subset of the initial dictionary considered by the GA when attempting to categorize documents from classes 1 and 9 (Commercial Banking and Soccer). The numbers under the class headings indicate how many documents from the test set contained each word. For instance, 19 of the 100 documents on commercial banking from the test set contained the word *advice*, while only one page on soccer contained that word. These are meant to give a coarse indication of the value of each word as a classifier. The words in bold are those included in the dictionary evolved by the GA. The majority of the chosen words are obviously useful classifiers and would likely have been selected by a human if they were hand-generating a dictionary. These include soccer terms and team locations such as *brazil, germany, stadium* and *match* and banking vocabulary including *credit, savings*, and *repayment*. A few words, such as *wednesday* appear to be useful classifiers but are unlikely to have been considered by a human because they are not strongly associated with the subject in our minds, even though they may appear frequently in documents on the subject. Some words that survived the evolutionary process are poor classifiers. Examples include *acorns* and *were*. It is possible that running the GA for more generations or with a larger population size could reduce the occurrence of these marginal choices. Alternatively, the fitness function parameters could be adjusted to introduce a larger penalty for dictionary size.

5.3. Prototype Evolution

The results of this experiment are shown in the fourth and fifth rows of Table II. In all cases the accuracy was improved over the unoptimized NNC while the number of prototypes was reduced by more than 50%. It should be noted that in order to achieve this result for classes 2 and 3, the fitness function parameter α had to be increased from 0.5 to 1. This removed the size penalty and thus allowed the GA to consider solutions involving more prototypes, which drove the accuracy higher even though the final solution does not contain an excessive number of prototypes.

The randomly generated prototype sets resulted in higher accuracy and fewer prototypes than the unoptimized NNC. Compared to the GA, however, the accuracy of the best randomly generated solution was significantly lower (5% lower for classes 1 and 9 and 9% for classes 2 and 3). The number of prototypes chosen was about the same in both cases. Unlike the feature selection case, the document distribution in the feature space seems to limit the combinations of prototypes that are capable of providing good results. The crossover operator is particularly important in this case because it allows useful subsets of prototypes to be combined in offspring.

Some characteristics of the evolved prototype sets are interesting. Table IV shows which documents were chosen as prototypes for each two-class trial. To keep the number of prototypes small enough to analyze easily, the size of the initial prototype set was limited to 15 documents from each class for this experiment (instead of the 50 used previously), and the test set was restricted to 30 documents of each type. The documents are numbered according to their class, so 1 through 1000 are class 1, 1001 through 2000 are class 2, and so on. In each case, the prototype set contains a roughly equal number of examples from each class being considered. The column labeled utilized shows how many times a given prototype was used to make a classification, and the accuracy column indicates how often that classification was correct. Ideally, the prototype set would contain a small number of frequently used prototypes that are reasonably accurate. If the feature space is such that there are several pockets of outliers among the documents to be classified, several prototypes may need to be included in the set in order to properly categorize these outliers. These prototypes would be utilized less often, but that would be acceptable if they were highly accurate. For classes 1 and 9, which are very dissimilar documents, the prototypes chosen display these traits. Documents 758, 550, 8668, 8452, and 8391 are used to make the majority of the classifications. Documents 550 and 8668 lead to some errors, but on the whole these prototypes are doing more good than harm. Documents 86, 578, and 8665 are only used a handful of times, but are perfectly accurate. In the other trial, which involves classifying more similar documents, the evolved prototype set is not as coherent. For instance, document 2347 is used to make one classification, and it was incorrect. Document 2309 is included in the prototype set despite not being used. It is clear that these prototypes cannot be helping the NNC. The GA could be "encouraged" to remove them by increasing the fitness penalty for the number of prototypes used. Alternatively, it could be forced to remove them by adding a specific check on the utilization versus accuracy of the prototypes in the fitness function. This ability to tailor solutions based on domain knowledge is one of the important benefits of a GA.

TABLE IV. Evolved Prototype Set

Classes 1 and 9			Classes 2 and 3		
Document	Utilized	Accuracy	Document	Utilized	Accuracy
86	1	1.0000	1901	5	1.0000
578	1	1.0000	1487	5	1.0000
758	19	1.0000	1455	2	.5000
550	8	.8750	1369	3	1.0000
8668	7	.7143	1038	2	1.0000
8452	6	1.0000	1540	3	.6667
8665	2	1.0000	2435	2	1.0000
8391	16	1.0000	2897	1	1.0000
			2310	28	.6071
			2141	6	1.0000
			2309	0	0
			2267	2	1.0000
			2347	1	0

5.4. Sequential vs Simultaneous Evolution

The results of this test are summarized in the last three rows of Table II. In each configuration, the accuracy of the evolved system was roughly the same as evolving the features or prototypes alone and represents a significant improvement over the unoptimized NNC. Furthermore, evolving both the feature and prototype sets resulted in about the same number of words and prototypes as when either was optimized individually. Most importantly, the nested genetic algorithm did not perform better than classifiers evolved sequentially. This is encouraging as it allows us to both improve the accuracy and decrease the computation time and memory requirements of the NNC using the much faster sequential algorithm. It is likely that the extremely high dimensionality of the feature space in text classification domains is the reason the feature and prototype sets are not so interrelated that they must be evolved simultaneously. The large number of features greatly increases the likelihood that documents will be nearly linearly separable, which simplifies the process of choosing prototypes. Our results indicate that it does not matter which is optimized first—the features or the prototypes. More work needs to be done in order to generalize these results beyond the particular document corpus studied here, however.

6. Conclusion

More research needs to be done to analyze the performance of other types of classifiers on this particular document corpus and on other text classification problems so that relative comparisons can be made. However, the results presented in the previous section do indicate that the NNC performs reasonably well in absolute terms in this domain. While the level of accuracy achieved by the NNC in this study was not high enough for applications where the consequences of a misclassification may be serious (e.g. spam filtering), there are potential uses. For example, the system's classification may be useful in optimizing searches for some domains, particularly web portals where the number of topics involved is naturally limited. If a user is looking for documents similar to the one they already have, this system can assist by directing searches to a particular category first. The search of this smaller space will be faster, and the documents are likely to be more relevant. If the desired information is not found by searching that group of documents, the search could be widened to consider all documents. We are considering this type of application as part of our future work in this area.

We have found genetic algorithms to be a useful method for optimizing the feature and prototype sets. Several studies in this area (using GA and other techniques) have reported that it was possible to reduce the feature set and the number of prototypes while maintaining performance, but that these optimizations did not increase accuracy ([9], [3]). Our results contradict this observation. They indicate that optimizing either the dictionary or the prototype set can increase accuracy, particularly in the easiest case of two very distinct subject areas. While random selection of the dictionary and prototypes did produce reasonable results, the ability of the GA to exploit useful subsets of features via the crossover operator proved useful in both cases. The option of using the fitness function to tailor results is also valuable. It can not be said that optimizing the dictionary is more important than reducing the prototype set or vice versa—the GA reduced both by about the same percentage, and the accuracy increased in both cases. Prototype evolution was complicated more by the similarity of the documents being considered than was dictionary evolution, but this problem can be mitigated by adjusting the fitness function parameters α and β, which will cause the system to work to achieve more accurate solutions at the expense of size and will penalize larger dictionaries more than larger prototype sets.

The computation time and memory requirements of an NNC scale with the product of the number of features and prototypes. In the studies cited, the numbers of features and prototypes were small so the potential savings from optimizations were limited. However, in the area of text classification the number of features can range from hundreds to thousands so any reduction of the feature or prototype sets translates into a significant improvement in computation time. Although optimizing both sets at once (sequentially or simultaneously) did not reduce the size of either set beyond evolution of either one in isolation, it did lead to better performance (because the sizes of both sets were reduced). Furthermore, the sequential evolution algorithm was shown to perform as well as the more computationally-intensive simultaneous evolution.

References

[1] M. Sinka and D. Corne, "Evolving Document Features for Web Document Clustering: a Feasibility Study," in *Proceedings of the IEEE Congress of Evolutionary Computation*, June 2004.

[2] Y. Yang and X. Liu, "A Re-Examination of Text Categorization Methods," in *Proceedings of the Special Interest Group on Information Retrieval*, 1999, pp. 42–49.

[3] L. Kuncheva and L. Jain, "Nearest Neighbor Classifier: Simultaneous Editing and Feature Selection," *Pattern Recognition Letters*, no. 20, 1999, pp. 1149–1156.

[4] P. Langley and W. Iba, "Average-Case Analysis of a Nearest Neighbor Algorithm," in *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 1993, pp. 889–894.

[5] I. Guyon and A. Elisseeff, "An Introduction to Variable and Feature Selection," *Journal of Machine Learning Research*, vol. 3, 2003, pp. 1157–1182.

[6] D. Zongker and A. Jain, "Algorithms for Feature Selection: an Evaluation," in *Proceedings of the International Conference on Pattern Recognition*, August 1996, pp. 18–22.

[7] S. Ho, C. Liu, and S. Liu, "Design of an Optimal Nearest Neighbor Classifier Using an Intelligent Genetic Algorithm," *Pattern Recognition Letters*, no. 23, 2002, pp. 1495–1503.

[8] L. Kuncheva and J. Bezdek, "Nearest Prototype Classifier: Clustering, Genetic Algorithms, or Random Search?" *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, vol. 28, no. 1, 1998, pp. 160–164.

[9] F. Brill, D. Brown, and W. Martin, "Fast Genetic Selection of Features for Neural Network Classifiers," *IEEE Transactions on Neural Networks*, vol. 3, no. 2, 1992, pp. 324–328.