Online Text Categorization using Genetic Algorithms

İ. E. Şahin
Bilkent University
Department of Computer Engineering
06800, Bilkent, Ankara, Turkey
iesahin@cs.bilkent.edu.tr

April 7, 2005

1 Introduction

The problem of text categorization is to associate texts with categories according to their semantic properties. Text categorization (a.k.a. Text Classification) problem with current ML solutions to it are described in [1]. In this work, my goal is to present a study of genetic algorithms in Text Categorization problem. GA are not one of the major techniques for TC and in [1], author only informs about the existence of such work, without giving any details about the research accumulated so far. For TC, there seems to be little interest apart from general classification problem.

In this work, I will give an algorithm that learns through applying user interaction as a fitness function for GA. Similar methods are applied to different problems before, like identification of faces. [2] Nevertheless, the algorithm presented in this paper can be used for batch processing through a set of classified text and user interaction is not necessary for training in that case. Algorithm exploits evolutionary nature of GA to provide a behavior that enables user interaction.

Text Categorization has its applications in document indexing, automated metadata generation, word sense disambiguation, building catalogs of Web resources and in general any application requiring document organization. It is a task to assign a document d_j to a category c_i (used with *class* interchangeably) by approximating a function $\hat{\Phi} : D \times C \rightarrow \{T, F\}$ by maximizing the coincidence of $\hat{\Phi}$ with the actual categorization, Φ, where $D = \{d_1, \ldots, d_n\}$ is the set of documents, $C = \{c_1, \ldots, c_m\}$ is the set of classes, and T and F are boolean values for true and false, respectively. In this problem, categories are just labels without any other declarative or procedural semantics and no *exogenous* information such as publication date,

document type, publication source etc. are assumed to be available. Categorization must depend solely on information extracted from document. [1]

A classifier may assign a document to multiple classes or a single class, depending on the constraints. A special case of single category classification is called *binary* categorization where a document d_j is assigned to a class c_i or its complement $\bar{c}_i$. Most of the TC literature is devoted to binary case, because of its simpler illustration, having direct applications such as filtering (e.g. spam filtering) and ability to generalize for multi-label case.[1] In this paper, I will also consider the binary case. Moreover, I assume that categories are populated more or less evenly for basic algorithm. Though there is an extension for uneven multi-category case, initial exposition of the algorithm is based on even distribution of documents between a category and its complement.

2 Algorithm

2.1 Decision and Basic Model

For the single category case, TC problem is to classify a document D as belonging to a class C or not. Let $G_C = \{g_1, \ldots, g_n\}$ be a set of chromosomes for class C. Each chromosome has a vote for D_j, either positive or negative about D_j's belonging to C . For n (number of chromosomes) being an odd number, majority on G_C is decisive for D_j belonging to class C.

Each chromosome g_i has a number of genes $\Gamma = \{\gamma_1, \ldots, \gamma_m\}$ to make a decision about D_j. γ_i consists of a word w_i unique for chromosome and a valency v_i that is either 1 or 0. For m (number of genes) being an odd number, majority of $R = \{r_1, \ldots, r_m\}$ determines the vote of g_i. r_i is determined with respect to a document D with the following rule:

- if D contains w_i and $v_i = 1$, $r_i = 1$
- if D does not contain w_i and $v_i = 0$, $r_i = 1$
- otherwise (if D contains but mustn't contain, or D doesn't contain but must contain), $r_i = 0$.

After G_C decided for classification, user interaction or previously classified data gives a correct classification for D, which is identical to decision made by G_C or not. This result is used as a fitness function for g_i to determine the next phase of population in G_C. g_i has power p_i and this power is increased or decreased by a factor f according to its position in voting.

$$p_i^{t+1} = \begin{cases} p_i^t + f & \text{if } R_i = R_{C,D} \\ p_i^t - f & \text{if } R_i \neq R_{C,D} \end{cases}$$

where p_i^t and p_i^{t+1} denote power in current and next phases respectively, R_i is the vote of g_i in classification process and $R_{C,D}$ is the correct classification for D, either positive or negative. There is no limit for f, but since mating probability is determined by p_i, it must be determined to reflect the sensitivity to errors in classification. An alternative is to have different f for correct and incorrect voting, namely f_+ and f_- to change p_i asymmetrically, however this is a minor variation. f can also be a function of p_i^t so that the change is nonlinear, but this is again a variation to be considered after initial tests.

After p_i is determined, mating between g_i is determined. In each turn, two randomly selected chromosome is mated. The probability to participate in mating for g_i is

$$\frac{p_i}{\sum_{j=1}^{n} p_j}$$

So, in the long run, a chromosome that votes for incorrect classification will not be able to mate.

2.2 Crossover

There are two elements of g_i to be crossed over with g_j, for i and j to be the indices of the mating elements selected. These are mating probabilities p_i and p_j, and genes Γ_i and Γ_j. For g_k is the chromosome produced by mating g_i and g_j,

$$p_k = \frac{p_i + p_j}{2}$$

and Γ_k is reproduced by following procedure:

Let $\Gamma_=$ be the set for identical elements in Γ_i and Γ_j.

$$\Gamma_= = \{\gamma_k | (\gamma_k \in \Gamma_i) \wedge (\gamma_k \in \Gamma_j)\}$$

Let $\Gamma_{i,\neq}$ be the set for elements in Γ_i which is not an element of Γ_j. $\Gamma_{j,\neq}$ is defined similarly.

$$\Gamma_{i,\neq} = \{\gamma_k | (\gamma_k \in \Gamma_i) \wedge (\gamma_k \notin \Gamma_j)\}$$

and

$$\Gamma_{j,\neq} = \{\gamma_k | (\gamma_k \in \Gamma_j) \wedge (\gamma_k \notin \Gamma_i)\}$$

Define Γ_k as

$$\Gamma_k = \Gamma_= \cup Z(\Gamma_{i,\neq}, \Gamma_{j,\neq})$$

where Z is a n-point crossover function that selects elements from either set and returns a set having a size $n - m$ where m is the number of identical elements in Γ_i and Γ_j.

This way removes the necessity to check for multiple elements in Γ_k and will make salient words that are common to all documents persistent. A

drawback to this scheme is the slowness to fix incorrect words common to all chromosome. This problem is parallel to local optima problem we face in all hill-climbing algorithms. A remedy for this case is mutation which is described in the next section.

2.3 Mutation

Mutation is primary means to learn new information in newly encountered documents. Unlike traditional GA, probability of mutation is higher and dependent to the number of documents encountered. If, in the beginning, $n = 5$ and a new document is encountered first time, $t = 6$ and probability of mutation $p_m = \frac{1}{t}$. t is the number of classifications plus number of chromosomes in the system. In the course of classification, as t increase, p_m will decrease and in the long run, G will be stabilized.

Let a document D consists of unique words $D_w = \{d_1, \ldots, d_n\}$. Let $R_{C,D}$ be the real classification (state of the world) of document D according to class C. $R_{C,D}$ determines the valence of d_j , a random element selected from D_w according to its frequency in the document. Frequent words appearing in the document will be likely to be selected for replacing words in the genes. For each $\gamma_i = (w_i, v_i) \in \Gamma$ of a chromosome γ_i is replaced by $(d_j, R_{C,D})$ with a probability of p_m, if d_j is not in words of Γalready.

In other words, if $R_{C,D}$ is positive, we will have new words in Γ with a positive valence ($v_i = 1$) and if $R_{C,D}$ is negative, we will have new words with $v_i = 0$.

2.4 Initialization

Initialization of classifier is similar to mutation. In this case, however, instead of replacing, mutation procedure selects words with a probability determined by their frequency in the document. Since there is no negative instance, all valence in this phase is 1.

Let there be a different document D_i for each chromosome in this phase. Let W_i be the unique words of this document and for each word w_i let there be a function $C_{D_i}(w)$ that returns the number of w in D_i. For m is the number of genes, we select a word w_i to build $\gamma_j = (w_i, 1)$ with a probability of

$$\frac{C_{D_s}(w_i)}{\sum_{k=1}^{n} C_{D_s}(w_k)}$$

where n is the cardinality of W_s and $C_{D_s}(w)$ is the number of w in document D_s. Note that, in each step, the selected word, w_i is removed from W_i so probability of each word changes. This scheme is identical to the basic mating procedure of GA.

2.5 Variations

There are some variations to the basic algorithm described above. These variations may give good results in some cases where the assumptions of the basic algorithm do not hold. Although there are no tests regarding these variations currently, as a future work, these variations may be tested and possible outcomes are reported on different datasets.

2.5.1 Including New Document Information as a Chromosome

Including newly acquired information by a new document is solved in basic algorithm with mutation. In that scheme, mutation elements are determined by new document and thus all chromosomes are affected from new information. Another way to include information from document is to create a brand new chromosome from document that has the same number of genes with other chromosomes and giving it a power proportional to its desired effect; making it to mate with other chromosomes as normal.

Like the initialization of chromosomes, if the document is a positive instance that belongs to the class, valence of its words in genes are all 1 and if the document is a negative instance, valence of its words in genes are all 0.

In this case, information from new document will affect a particular chromosome if it has any chance to affect. The proposed voting and crossover scheme does not allow more than 50% change in genes in one crossover. Since there is no control over mating except power, new document's probability to affect voting scheme seems to be limited than that of mutation way. Along with this, without elitism, chromosomes representing different local optima may yield poor results. In mutation, a newcomer's effect is limited in especially later cases, however in this case if it is able to mate, it can alter behavior of resulting chromosome at once.

All these disadvantageous cases may be read advantageous if new document is a good representative that may feed good results in later generations of chromosomes.

2.5.2 Multiple Classes and Mutation

Despite the algorithm's suitability for multiple classes by introducing different chromosomes for each class, mutation model tends to increase negative elements (elements that have $v = 0$ in genes) in chromosomes as the number of classes increase. The reason behind this is to allocate more training to negative instances, due to increased relative frequency of documents for a particular class. For single class case, the number of negative instances can be assumed to be around positive instances. However as the ratio of documents in a particular class decrease, this assumption (despite its inherent unreliability in the beginning) becomes more and more invalid and genes in chromosomes begin to be shaped by elements outside of the class and

this possibly decreases the performance in real world applications. Learning what class is *not* cannot be relied on for classification because practically, there may be infinite number of words that does not represent a class. And classifying a document because of the lack of a finite subset of this infinite words guarantees a failure in classification.

In order to solve this, an option is to eliminate all negative learning from the algorithm and relying only on the representative words of class. However, for textual information where nuances between classes are not always shown by representatives of classes and where negation is sometimes necessary for definitions, this scheme will perhaps yield a poor general performance too.

Another option is to make negative learning harder by reducing mutation probability. A factor that can help in this option is the prior probabilities of classes. In mutation of chromosomes (§2.3) p_m for negative instances multiplied by $P(C_i)$, the prior probability of class before determining a possible mutation in genes. This way, the probability to have 0 in valence field of genes is reduced by the ratio of documents in this class to the ratio of elements in total.

In current tests, where a single class is used with a prior probability of 0.5, this modification is not reflected. Nevertheless, this is an important modification for future versions to be tested with multiple classes.

2.5.3 Elitism

The basic algorithm does not include a version of elitism to conserve high performance chromosomes. The hypothesis behind this says: After processing a high number of documents, chromosomes will be stable enough to not to preserve them against defects in the documents. Intuitively, there will be representative words twice the number of genes in a chromosome after some time. However, this may not give good performance and may result in low accuracy at the end; which is unlikely to be recovered.

An alleviation to this problem is to have some number of chromosomes in a reserved place for each class as the elite ones. Since we have a metric, power, in determination of the quality of chromosomes; there can be some parking place for those chromosomes that have the highest ranking in classification in the course training.

However, due to difference of textual data, success of classification may not be repeatable. Since the algorithm may change the qualia of a class during training, success at some training time t may not represent the actual success of chromosome. For textual classification, it is harder to propose a consistent scheme for success of classifier since the classification process is about semantics of the document and there is no way to enumerate semantics which allows an objective success.

Another use for elitism is building a caste instead of reducing effect of later documents. In this case, training does not change the behavior of de-

cision making chromosomes immediately. Instead a lower level of "common" chromosomes are maintained along with elites and effect of new training documents are only applied to lower level chromosomes. Actual voting is carried out by elite ones, but evaluation of common ones still continue. If any of the commons is able to maintain a higher power than an elite chromosome, these are exchanged. In this scheme, an oligarchy of chromosomes

3 Evaluation

The implementation of the algorithm is written in Python. In this decision, language's capabilities regarding string processing played a role. As a dataset, I used Reuters-21578 data [3] since it is a standard among text categorization community and has a size that makes managing and evaluation easier.

I implemented an SGML parser to parse Reuters data files. Reuters-21578 data has 27244419 total and 72967 distinct words. From these, I have filtered the most frequent 200 words that make up 14066768 (~51%) words in the corpus. I have also removed words containing digits and all punctuation from the dataset. All textual data is converted to lowercase. A naive stemming algorithm which drops all final "es", "s", "ed" and "ing" from words is applied to the dataset.

In voting, thresholds are set to $\frac{1}{3}$ of total chromosomes for classifier and $\frac{1}{3}$ of total genes in chromosome. This means, for any chromosome to vote for the document, $\frac{1}{3}$ of the genes must be in coherence with the document. For a text to be classifier, at least $\frac{1}{3}$ of the chromosomes must decide that this text belongs to this class.

3.1 Tests for number of chromosomes and genes

I tested the parameters, number of chromosomes per class and number of genes per chromosome. For this test, all 1912 instances from Reuters-21578 "coffee" category is applied to the classifier with the same number of random texts from other categories.

In Table 1 accuracy of the classifier is maximized when n (number of chromosomes) is equal to 7 or 9. In this case, for 11 genes, there are 77 or 99 total features for the category are considered for classification. Note that, beyond this, accuracy decreases about 3% per 100 feature and this shows the role of overfitting for the classification clearly. An interesting point in the results is the 1-2% consistent difference between training and test data. As we have mentioned, while training, features are also crossed over and mutation from current data is applied for the next generation. However, this scheme does not yield an optimum performance and with the same parameters, in almost all cases, a classifier that does no training while

Table 1: Accuracy of Classifier for "coffee" category, number of genes = 11

Number of chromosomes	Accuracy Training Data	Accuracy Test Data
1	0.5	0.5
3	0.69	0.68
5	0.71	0.72
7	0.75	0.75
9	0.73	0.75
11	0.71	0.73
13	0.73	0.75
15	0.72	0.73
17	0.73	0.75
19	0.69	0.71
21	0.69	0.75
29	0.70	0.74
31	0.69	0.71
33	0.67	0.72
35	0.69	0.72
39	0.73	0.72
49	0.66	0.73
59	0.71	0.73
69	0.67	0.69
89	0.66	0.71
111	0.66	0.71

classification gives better results. Reasons behind this must be investigated thoroughly in the future.

In Table 2, same category is used to evaluate the threshold to set for number of genes per chromosome. As it can be seen from the table, there is a steady decrease in the accuracy in training phase while it is oscillating around 67% for the test data.

3.2 Effect of Data Set Size

One of the big problems is having large number of documents to classify, while rather dull tools to describe the semantics of categories. Examples are "palladium", "platinum" and "silver" categories of the dataset. In these, I had high rates of accuracy over less number of documents (48 for palladium, 132 for platinum, 384 for silver) given 11 genes in each of 7 chromosomes. Results for these conditions are in Table

Table 2: Accuracy of Classifier for "coffee" category, number of Chromosomes = 7

Genes	Accuracy Training	Accuracy Test
11	0.75	0.75
15	0.64	0.67
19	0.64	0.68
21	0.65	0.67
25	0.61	0.66
29	0.61	0.68

Table 3: Sensitivity to Data Set Size, 11 genes, 7 chromosomes

Category	Train Acc.	Test Acc.
Palladium (48)	0.94	0.96
Platinum (132)	0.64	0.81
Silver (384)	0.79	0.78
Coffee (1912)	0.75	0.75

The reason behind this sensitivity to data set size is the increased false negatives in outcomes. For example, in "coffee" data set's testing results, there are 440 rejected true documents, while 517 are accepted. The number of false positives for this case is only 32 while 925 elements are correctly identified as alien to the category. The increase in dataset size triggers the increase in number of false negatives more than others. For a document to be classified in a category, it must have $\frac{77}{9} \approx 8$ words in common with chromosomes in the configuration giving this results. However, as we have seen, requiring more number of words does not lead a sharp drop in accuracy.

Crossing over words and expecting a mixture of document structure and category semantics is, in fact, a bizarre way of applying statistics to semantics, though current techniques to text categorization do the same thing, maybe with some extra care and information. Above figures show that, for large datasets, it is hard to expect a correct classification using only word lists. There is perhaps some way to discriminate quality of information for a given dataset other than just counting the number of words. There must be a hierarchy from the most salient words of the category to the least, while preserving nuances between categories.

4 Future Work and Conclusion

This work is a chopped down version of a greater one, where the structural properties of document are also considered and all documents are specified with a tree and category can be specified with crossovers and mutations over this tree. Representing salient information of a document as a tree where the root is the most important element of document can lead to a tree where the semantics of a category is represented through relations between words rather than numbers. This way, there could be a better representation of semantics and errors leading to false classification can be overcome by relations of this type.

Despite the high error rate in classification, this work told me where does a "word list" approach to text categorization may fail. There is no "right number of words" to classify a document into a category, since increasing the number of words also increases the number of inconsistencies between elements of classifier and document. "Word lists" which are used in almost all Machine Learning approaches to Text Categorization, simply result in a very rough shape that must be interpreted with a very skilled machine. Since we cannot adjust the intelligence provided by computers, our only exit is to adjust the representation we use for information. A natural one which leads to learning the world as the problem progresses; rather than association of statistics can yield better results in terms of intelligence.

With these ideals, there can be much to be done for this work to become a real classifier. Genes having a valence of 0 must be analyzed thoroughly, because their benefit may not deserve the complexity they bring. Also for multi-category case, where the probability of mutation must adjusted, there can be a better scheme to include new information as it becomes available.

The holy grail of semantics in computers is to describe the world to computer as we describe it to another human. This ideal seems to be impossible to achieve by just symbol manipulation, because "meaning" for a human is not sourced directly from symbols. Without relating symbols to perception, there is possibly no way that can make to tell the meaning in all facets. We are like in front of a person without any senses, no vision, no touching, no smelling while we are trying to tell a machine how the world works. Text Categorization, with its all broad coverage of problems that we face and solve daily, is a good representative of this kind of problems.

References

[1] Sebastiani, F.; Machine Learning in Automated Text Categorization; ACM Computing Surveys, Volume 34, Issue 1, March 2002 http://doi.acm.org/10.1145/505282.505283

[2] Goldberg, David; Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, 1989

[3] Hettich, S., Bay, S. D. (1999). Reuters-21578 Dataset in The UCI KDD Archive, Irvine, CA: University of California, Department of Information and Computer Science http://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html

[4] Beasley D., Bull D. R. Martin R. R., An Overview of Genetic Algorithms: Part 1 Fundamentals, University Computing, 1993 http://laseeb.isr.ist.utl.pt/tfc_dae/docs/ga_overview1.pdf

[5] Beasley D., Bull D. R. Martin R. R., An Overview of Genetic Algorithms: Part 2 Research Topics, University Computing, 1993 http://laseeb.isr.ist.utl.pt/tfc_dae/docs/ga_overview2.pdf

[6] Davis L., Steenstrup M., Genetic Algorithms and Simulated Annealing, Pitman, London, 1987

[7] Russell. S., Norvig P.; Artificial Intelligence: A Modern Approach; Prentice Hall, 2003