Kernel-based text-categorization

Olivier Teytaud, Radwan Jalam
ERIC, Université Lumière Lyon 2
5, avenue Pierre Mendès-France, F-69676 Bron cedex
{*jalam,oteytaud,aelissee*}*@eric.univ-lyon2.fr*

Abstract

This paper presents some techniques in text categorization. New algorithms, in particular a new SVM kernel for text categorization, are developed and compared to usual techniques. This kernel leads to a more natural space for elaborating separations than the euclidian space of frequencies or even inverse frequencies, as the distance in this space is the most usual distance between distributions. We give an application to the recognition of the author of a text, showing that text-categorization, after projection in this space, can be applied to quite subtle categorizations, and put into relief that our kernel could be used for any classification of distributions. We discuss the efficiency of our algorithms, depending upon the precision of the estimation of frequencies.

1 Introduction

Being given Q classes of texts, we call **text categorization** the task of determining the class of T, T being a text, after learning on a labelled training set. This can include language recognition, or topic recognition. We have restricted our study to algorithms using N-grams, because of their generality (they could be used for any kind of sequences on a discrete alphabet - see for example applications in biology) and their efficiency; we do not work on approaches based on dictionnaries. The most usual methods are 1-NN with dissimilarity measures, and [8] or [12] conclude (roughly) that the most efficient method is SVMs. We confirm these comparisons and introduce new techniques, based upon a new kernel.
Definitions: A being an alphabet, a **N-gram** is a sequence of N elements of A. For $N = 1$, a N-gram is a **letter**; 2-grams are called **bigrams**, 3-grams are called **trigrams**. The set of **words** is the set of all N-grams for any N. One calls *N-profile* of a family of texts the sequence of the N-grams of this family of texts, in decreasing order of frequency, with their frequencies.

2 Text categorization

2.1 With (dis)similarity measures

Many algorithms used for text categorization are based on distances or more generally on similarities and dissimilarities. All these methods rely in finding the closest

texts in the learning set from the one whose class is to be computed. The difficulty in this k-nearest neighbours approach is the definition of a distance. Indeed, one usually uses pseudo-distances. The simplest and oldest consists in building the profiles of each class and of the text, and then using the dissimilarity measure CT used by Cavnar and Trenkle in [2]: The distance between the two profiles P_1 and P_2 is defined as

$$CT(P_1, P_2) = \sum_{w \in P_1, R_{P_1}(w) < NMAX} min(|R_{P_2}(w) - R_{P_1}(w)|, DMAX)$$

where $|x|$ denotes the absolute value of x and $R_P(w)$, with w a N-gram and P a N-profile, denotes the rank of w in the profile P, if w belongs to P, and $DMAX$ otherwise (e.g., $NMAX = 500$ and $DMAX = 1000$). Another possible "distance" is the Kullbach-Leibler (KL) ([10]) dissimilarity measure:

$$KL(T_1, T_2) = \sum_{N_g} f_2(N_g) log(\frac{f_2(N_g)}{f_1(N_g)})$$

where the sum is taken over all N-grams.
with T_1 and T_2 some texts, and $f_i(N_g)$ the frequency of the N-gram N_g in the text T_i, plus half of the frequency of a N-gram which would occure once if N_g has frequency 0 in T_i (to avoid too much strong penalization of unseen N-grams).

Another possibility is the cosine dissimilarity measure ([7] uses a centered space on the mean of the frequency vectors; we here do not use this translation). This is the following:

$$COS(T_1, T_2) = 1 - \frac{\sum_{N_g} f_1(N_g) f_2(N_g)}{\sqrt{(\sum_{N_g} f_1(N_g)^2) \times (\sum_{N_g} f_2(N_g)^2)}}$$

We chose another dissimilarity measure, the χ^2 dissimilarity. This is the following:

$$\chi^2(T_1, T_2) = \sum_{N_g} \frac{(f_1(N_g) - f_2(N_g))^2}{f_2(N_g)}$$

One can symmetrize this "distance" by using $\chi^2(T_1, T_2) = 2\frac{(f_1(N_g) - f_2(N_g))^2}{f_1(N_g) + f_2(N_g)}$. We do this in our practical experiments. When $f_1(N_g)$ and $f_2(N_g)$ are 0, then we replace $\frac{(f_1(N_g) - f_2(N_g))^2}{f_1(N_g) + f_2(N_g)}$ by 0 (which is its continuous extension).

2.2 Classification methodes based upon an encoding in $\mathbb{R}^n$

Another approach consists in encoding documents by vectors, in order to classify points in $\mathbb{R}^n$. This allows the use of all classical methods: backpropagation neural networks, support vector machines (SVMs), k nearest neighbours in $\mathbb{R}^n$ (what can be done directly with the previous dissimilarity measures, too), decision trees... One has to choose an encoding, which can be used with both the training set and the test set; eg, let w_1,...,w_q be a finite set of words, and let's define x_i as the number of occurings of w_i in T (or its frequency). x will be the vector associated with T. The finite set of words can indeed be the set of all the *words* included in the considered texts, or the set of all N-grams for a given N. This number α of occurings can be replaced by different functions of α; [9] lists different possibilities. It's possible to consider only significant variables among all these ones. Different solutions are possible, among which, for this kind of data, the most famous is likely the information gain criterion (see [13]). Experimental results from [8] show that as much as possible, we must keep all the variables - what will be done in the sequel.

3 A new positive definite kernel for SVM ?

Encoding in $\mathbb{R}^n$ allows the use of lots of training algorithms, and in particular SVMs (see [11]). But one can use SVMs in another way: we define $K(T_1, T_2) = exp(-d(T_1, T_2))$, with d one of the dissimilarity measures suggested above. We experimented $K(T_1, T_2) = exp(-\frac{\chi^2(T_1,T_2)}{\sigma^2})$.

Conjecture: The function $k(T_1, T_2) = exp(-\frac{\chi^2(T_1,T_2)}{\sigma^2})$ is a positive definite kernel.

We so have a new kernel at our disposal, which has the following advantages:

- distance is "natural"; whereas with linear SVMs distance is the euclidian distance in the space of frequencies (or inverse frequencies), we look for RBF separations in a space with a classical distance among distributions.
- we can learn on a compact representation of datas - a kernel matrix $m \times m$, with m the number of texts in the training set.
- the hyperparameter σ is very easily chosen, as explained below.

4 How to use RBF networks for text categorization

As in the case of SVM, one can use an RBF network with the encoding of texts in $\mathbb{R}^n$; but one can use the χ^2 dissimilarity for example. As explained above, this corresponds to a linear separation in a feature space. This method is successfully tested below. The algorithm is summerized below, with (T_i) the family of labelled texts (used for training), (T'_i) the family of texts to be classified:

1. Let O be a matrix such that $O_{i,j} = 1$ if T_i belongs to class j, -1 otherwise else.
2. Let K be the matrix such that $K_{i,j} = exp(-\frac{\chi^2(T_i,T_j)}{\sigma^2})$ and K_1 the matrix resulting of K by adjunction of a column of 1's at its right.
3. Let K' be the matrices such that $K'_{i,j} = exp(-\frac{\chi^2(T'_i,T_j)}{\sigma^2})$ and K'_1 the matrix resulting of K' by adjunction of a column of 1's at its right.
4. Let W be the weight matrix such that $K_1 \times W = O$, let $O' = K'_1 \times W$.
5. We classify T'_i in class $argmax_k\ O'(i, k)$.

[8] explains (partly) the good behavior of SVMs on text categorization by its capacity to treat so many dimensions without having to select relevant variables. One can notice that RBF, with this particular kernel, verifies the same property: the training set is translated into a kernel matrix of size $m \times m$, taking into account *all* the information, with m the number of texts in the learning set.

The difference with the previous SVM algorithm is that with RBF there's no reason for W to be sparse. In the case of text-categorization, as the training set, whenever it is very big, leads to a little number of examples, one can suppose that this is not really a problem. SVM have the advantage of maximizing the margin, but they have one more hyperparameter, the constant C of penalization of errors.

5 Writer recognition: working on large samples

The success rate is evaluated by leave-one-out in the case of author recognition, as the training set is small (28 classes (= authors), 130 texts).

We use a set of french books (130), written by well known writers, like Balzac, Bloy, Corneille, Diderot, Engels, Flaubert, Fourier, France, Gaberel, Gautier, Gobineau, Hugo, Huysmans, Lamartine, Leibnitz, Maistre, Maupassant, Moliere, Pascal, Racine, Renard, Rostand, Rousseau, Sand, Stendhal, Verne, Voltaire, Zola. Some of this writers are translated from other languages. The complete list of titles is too much long for beeing listed here, but the used files can be asked by email to the authors. The fact that texts are not all formatted the same way hasn't been corrected, and is considered as a supplementary difficulty for the algorithm. Most of these texts come from the ABU site, cedric.cnam.fr/ABU/, the others from the Bibliothèque Nationale de France, www.bnf.fr/. The experimental results are the following ones (with 3-grams):

Algorithm	Success Rate
RBF with $(\chi^2)^p$ kernel for $p = \frac{1}{2}, \frac{1}{4}, ...\frac{1}{32}$	87.69 %
RBF with χ^2 kernel	86.15 %
Multiclass svm with χ^2 kernel	86.15 %
Multiclass linear svm	78.462 %
SVM with χ^2 kernel	72.3077 %
1-NN with χ^2 dissimilarity	70.77 %
linear SVM	67.69 %
1-NN with KL dissimilarity	52.31 %

All our tests are made with implementations in Octave (see www.che.wisc.edu/octave for a description of this very interesting free clone of Matlab). All the source codes can be asked by email to the authors. We call "multiclass SVM" a SVM designed for multiclass categorization, defined in [6]. It is worth puting into relief that in this case (high dimensionality, 28 classes) this SVM is significantly better than the usual method consisting in combining SVMs one-against-all as suggested in [11]. We have both SVM multiclass with χ^2 significantly better than SVM with χ^2 and linear SVM multiclass significantly better than linear SVM.
Our experiments gives the following results, with >> denoting a difference with confidence 5 %, $\geq$ a difference with confidence 15 %:

$$\{ \text{RBF - SVM Multiclass } (\chi^2) \} >> \text{SVM Multiclass} \geq \text{SVM } \chi^2 \text{ - SVM - 1-NN}$$

One can notice that our experiments, as the ones of [12], concern sequences large enough for a nice approximation of frequencies. The following experiments will be done on another case.

6 Language recognition: working on small samples

In this case the success rate is evaluated by validation on a disjoint part of the dataset. After the previous benchmark, one could conclude (too quickly) that RBF with χ^2 kernel seems to be the ultimate algorithm for text categorization. The multiclass version of SVMs looks as powerfull as it, but RBF are much faster and simpler to implement. In our following experiments, we will focus on two algorithms: RBF, because of their efficiency shown in the previous benchmark, and 1-NN, because of its simplicity, efficiency in the following case as we will see in the experiments below, and because it's widely used in practical applications. The following experiments are made with Java implementations, based on the Jama matrix package. All java source codes can be asked by email to the authors, or found at URL eric.univ-lyon2.fr/~jalam/java/devineur. The task consists in recognizing in

which language is written a given text. We work on five languages: french, arabic, english, spanish and german. As this is known a very easy task, we complicate it by using very small parts of texts. We detail a comparison on a particular set of 250 samples of 100 bytes, then 500 samples of 50 bytes, then 1250 samples of 20 bytes (20 bytes on average). We have 5 big texts of 5 Ko used to define profiles (come from G. van Noord's page), and short samples from 5 languages (arabic ones built with html pages, german ones from "Stochastic Language Identifier" (www.dougb.com), french ones from a book at www.alyon.org, english and spanish ones from the corpus of [4]). All the used datasets can be asked by email to the authors. With a testing set made of samples of 100, 50 or 20 bytes, SR meaning "success rate":

Algorithm	SR (100 bytes)	SR (50 bytes)	SR (20 bytes)
1-NN (KL)	100 %	99.4 %	92.8 %
1-NN (χ^2)	98.8 %	96.6 %	87.92 %
RBF ($\sigma^2 = 10$)	37.6 % (100 %)		
RBF ($\sigma^2 = 100$)	98.8 %	93 %	71.04 %

The result between parenthesis is got with profiles computed on 50 subparts of the training set instead of one profile computed on the whole training set (per class). This leads to better results for some RBF learnings - this trick doesn't work as well for the experiments with shorter samples. We now work with 250 samples of 100 bytes as learning set, to study more precisely the influence of "gathering" learning texts for RBF or k-NN :

Algorithm	Hyperparameters	SR (100)	SR (50)	SR (20)
RBF	$\sigma^2 = 10$	99.2 %	84.8 %	31.52 %
	$\sigma^2 = 100$	98 %	93.2 %	71 %
RBF (2-gathered prof.)	$\sigma^2 = 100$	97.2 %	88 %	68.56 %
RBF (5-gathered prof.)	$\sigma^2 = 1000$	98.8 %	94 %	80.88 %
RBF (10-gathered prof.)	$\sigma^2 = 1000$	99.2 %	95.2 %	76.72 %
RBF (25-gathered prof.)	$\sigma^2 = 1000$	98.8 %	94.8 %	82.4 %
RBF (gathered prof.)	$\sigma^2 = 100$	88.4 %	80.6 %	
	$\sigma^2 = 100000$	87.6%	77.4 %	61.36 %
1-NN	χ^2	99.2 %	96.6 %	88.4 %
1-NN	KL			47.2 %
1-NN (2-gathered prof.)	χ^2	99.6 %	96.8 %	88.8 %
1-NN (5-gathered prof.)	χ^2	100 %	97.6 %	90 %
1-NN (10-gathered prof.)	χ^2	99.2 %	97.2 %	88.56 %
1-NN (10-gathered prof.)	KL			89.84 %
1-NN (25-gathered prof.)	χ^2	100 %	96.8 %	87.2 %
1-NN (gathered prof.)	χ^2	100 %	93 %	84.56 %
1-NN (gathered prof.)	KL	99.7 %	97.4 %	89.4 %

"m-gathered profiles" means that the training texts have been gathered m by m; "gathered", that all texts of a class in the training set have been gathered. Keeping m small preserves the variability of the training set, m larger leads to more well defined profiles. The fact that m larger increases the efficiency of RBF in the case of very short texts, suggests that RBF could work here only as an approximation of nearest neighbours. In the case of small testing samples, KL remains better than χ^2, but KL seems to be unable to work with short learning samples, as illustrated by the case of non-gathered learning samples - this could be a problem for other tasks.

The hyperparameter σ^2 for RBF-learning was very easily chosen in the previous benchmark (classification by authors), as the success rate was constant for a wide range of σ and as empirical success was closely related to generalization success; *but in the case of 20 bytes strings*, the efficiency was very depending on σ and on the gathering; this leads to **two** difficult hyperparameters.

7 Conclusion

On datasets for which all frequencies are well defined (what doesn't mean that they only depend upon the class - they depend upon the author, the language, the topic, the time of the writing...), one can finally sum up previous results ([12], [8]) and our results by:

$$\text{RBF} > \text{SVM Mc } (\chi^2) > \text{SVM Mc} > \text{SVM } (\chi^2) > \text{SVM} > \text{1-NN} > \text{LLSF, C4.5, NNets} > \text{NB}$$

With SVM Mc the multiclass SVM from [6], SVM being a classical one-against-all SVM, LLSF as described in [12], NNets being neural nets other than SVM, C4.5 being the most famous algorithm of induction trees (see [3] for a use in text categorization) and NB being the Naive Bayes algorithm (see [5]). Notice that RBF > SVM Mc is not significant in terms of performance; we keep this comparison as RBF have the advantage of being much faster for learning and much easier to implement. The good results resulting from linear separations in the Reproducing Kernel Hilbert Space associated to our symmetrized χ^2 distance suggests that this space is the natural place where one can study separations between classes of distributions.
In the case of less-well defined frequencies (with very small parts of text in the testing set), 1-NN becomes better than RBF, with KL > χ^2 provided that the learning set is large enough to well define frequencies. The results of [4] with Markov Models, with two languages instead of five here, compared with our results, suggest that Markov Models trained with $25Ko$ per language have nearly the same error rate than 1-NN with $5Ko$ per language - random error rate being 20% with 5 languages and 50% with 2 languages, 1-NN seem to be more adapted to this task than Markov Models. Our tested version of 1-NN uses 3-grams, as Markov models of order 2 (which are often the most efficient according to [4]); 1-NN do not require computations of bigger profiles than Markov Models. Moreover, k-NN can efficiently work only keeping one profile per class, what is not always true with RBF; k-NN have the advantage of robustness (any gathering of profiles, no hyperparameter) - so we make the assumption that 1-NN and more generally k-NN are the most efficient solution to classify small samples of texts. The choice of the distance is an interesting question; because the dissimilarity CT isn't mathematically justified, and because the KL measure has difficulties for small learning samples (it implies particular cases for unseen N-grams and has an experimental bad behaviour on small samples...) we prefer the χ^2 dissimilarity, which didn't give significantly worst results than other distances (KL, CT or cosine) with well-defined probabilities and sometimes much better ones; but we recall that for small testing sets KL gave the best results. The experiments of [10] confirm this point. Finally, we underline that a detailed study shows that for most of our algorithms errors come from unbalanced classifiers (ie one class is "invading" the others). This suggests that algorithms "helping" handicaped classes (typically boosting) could give good results. Perhaps in a future paper boosted RBF/SVM > RBF/SVM...

We thank André Elisseeff for the multiclass SVM and for fruitfull discussions, regretting that he considered his contribution to this work too small for being author.

References

[1] C. Berg, J.-P.-R. Christensen, P. Ressel, *Harmonic Analysis on Semigroups, Theory of Positive Definite and Related Functions, Springer, 1984*

[2] W.-B. Cavnar, J.-M. Trenkle, *N-gram Based text categorization. In 1994 Symposium on Document Analysis and Information Retrieval in Las Vegas, 1994*

[3] S.-L. Crawford, R.-M. Fung, L.-A. Appelbaum, R.-M. Tong, *Classification trees for information retrieval, in Machine Learning: proceedings of the eighth International Workshop (1991), Morgan Kaufmann, pp 245-249*

[4] T. Dunning, *Statistical Identification of languages, Computing Research Laboratory Technical Memo MCCS 94-273, New Mexico State University, Las Cruces, New Mexico, 1994*

[5] I.-J. Good, *The estimation of probabilities: An Essay on Modern Bayesian Methods, MIT Press, 1965*

[6] Y. Guermeur, A. Elisseeff, H. Paugam-Moisy, *A new multiclass SVM based on a uniform convergence result. accepted at IJCNN'2000*

[7] S. Huffman, *Acquaintance: Language-Independent Document Categorization by N-Grams, in TREC 4 Proceedings, 1996*

[8] T. Joachims, *Text Categorization with Support Vector Machines: Learning with Many Relevant Features, in Machine Learning: ECML-98, Tenth European Conference on Machine Learning, pp. 137–142, 1998*

[9] Mehran Sahami, *Thesis:Using Machine Learning to Improve Information Access, Ph.D. in Computer Science, Stanford University, 1999*

[10] P. Sibun, J.C. Reynar, *Language identification: Examining the issues. In Symposium on Document Analysis and Information Retrieval, pp. 125–135, Las Vegas, 1996*

[11] V.N. Vapnik, *The Nature of Statistical Learning, Springer, 1995*

[12] Y. Yang, X. Liu, *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 15-19, 1999, Berkeley, CA, USA. ACM, 1999*

[13] Y. Yang, J. Pedersen, *A comparative study on feature selection in text categorization, in International Conference on Machine Learning (ICML), 1997*