Encoding Ordinal Features into Binary Features for Text Classification

Andrea Esuli and Fabrizio Sebastiani

Istituto di Scienza e Tecnologia dell'Informazione
Consiglio Nazionale delle Ricerche
Via Giuseppe Moruzzi 1 – 56124 Pisa, Italy
`firstname.lastname@isti.cnr.it`

Abstract. We propose a method by means of which supervised learning algorithms that only accept binary input can be extended to use ordinal (i.e., integer-valued) input. This is much needed in text classification, since it becomes thus possible to endow these learning devices with term frequency information, rather than just information on the presence/absence of the term in the document. We test two different learners based on "boosting", and show that the use of our method allows them to obtain effectiveness gains. We also show that one of these boosting methods, once endowed with the representations generated by our method, outperforms an SVM learner with tfidf-weighted input.

1 Introduction

In text classification (TC) and other IR applications involving supervised learning, the decision as to whether documents should be represented by binary vectors or by weighted vectors essentially depends on the learning algorithm used. If the learning algorithm accepts real-valued vectors as input, then weighted representations are typically used, since they are known to represent the semantics of the documents more faithfully, and to bring about higher effectiveness. On the contrary, some supervised learning algorithms (such as, e.g., most naïve Bayesian probabilistic learners, most learners based on decision trees, and most "boosting"-based algorithms) require binary input. This is a big limitation for these algorithms, since the information inherent in the statistical distribution of terms within the document (*term frequency*, aka *within-document frequency*) and within the collection (*document frequency*) is lost.

This is particularly frustrating in the case of boosting algorithms (e.g., AdaBoost.MH [1]), since their effectiveness rivals that of other state-of-the-art algorithms, such as support vector machines and other kernel methods, that can indeed use weighted features. What levels of effectiveness could boosting methods achieve if they could avail themselves of information deriving from the distribution of features within the document and within the collection? To be fair, it should be mentioned that AdaBoost.MH and related methods, while accepting binary input only, *do* use information from the distribution of terms *within the collection* (and across the labels), since this information is learnt from the training data and used internally, in order to pick the “pivot term” on which the learner built at the current iteration of the boosting process hinges. However, information from the distribution of terms within the document is not used in any form by these algorithms. And one hint that this information is indeed important for text classification comes by comparing the Bernoulli and the multinomial versions of the naïve Bayes method: the latter, which uses term frequency information, has been shown to substantially outperform the former, which does not make use of such information [2]. This paper gives an initial contribution to the solution of this problem by presenting a way in which term frequency can be indeed exploited in learning algorithms that accept binary input only.

The rest of the paper is structured as follows. Section 2 presents our solution to the problem of allowing learning algorithms that only accept binary input to use term-frequency information. In Section 3 we turn our attention to describing the experiments we have done and the results we have obtained, while in Section 4 we discuss related work.

2 Turning Ordinal Features into Binary Features

Our solution to the problem of allowing learning algorithms that only accept binary input to use term-frequency information, may be seen as extending these algorithms in a way that allows them to accept, as input, vectors of *ordinal* (i.e., integer-valued) features; this will make it possible to use the number of occurrences of term t_k in document d_j (noted $\#(t_k, d_j)$) as the weight of t_k for d_j. In a nutshell, our solution consists in encoding ordinal features as binary features, so that these algorithms can be applied unchanged to the representation generated by this encoding. If we view a binary feature as a binary-valued constraint of type $\#(t_k, -) \geq 1$, an ordinal feature can be represented by several binary-valued constraints of type $\#(t_k, -) \geq n$, where $n \in \mathbb{N}$. Given that these constraints are also binary-valued, they can be used as features for learning algorithms requiring binary input.

In principle this solution might seem infeasible, since it seems that a countably infinite number of constraints need to be generated for each feature t_k. However, this is not true in practice, since the number of constraints that need to be generated for each feature t_k is limited by the number of different nonzero values that $\#(t_k, -)$ takes in the training set. For instance, assume that t_k occurs twice is a few training documents, once in a few others, and zero in the rest of the training set. Clearly, there is no value in having a constraint of type $\#(t_k, -) \geq n$ for any $n \geq 3$, since this constraint is not satisfied by any example in the training set, and is thus akin, in standard bag-of-words representation, to a term that never occurs in the training set.

Our solution thus consists in generating, for all features t_k of the original binary representation, and for all *and only* the different values $v_1, \ldots, v_{n(k)}$ that $\#(t_k, -)$ takes in the training set, a binary feature t_k^x defined as $\#(t_k, -) \geq v_x$, for $x \in \{1, \ldots, n(k)\}$.

Finally, note that it might seem, at first sight, that this solution only allows using the form of term frequency consisting of the raw number of occurrences of the term in the document, i.e., $tf(t_k, d_j) = \#(t_k, d_j)$, thus preventing the use of more commonly used, real-valued forms such as, e.g., $tf(t_k, d_j) = \log(1 + \#(t_k, d_j))$. In practice this is a non-issue, since all forms that tf has taken in the literature are monotonically non-decreasing, and are thus equivalent from our viewpoint; that is, we might equivalently generate binary features t_k^x defined as $\log(1 + \#(t_k, d_j)) \geq \log(1 + v_x)$, for $x \in \{1, \ldots, n(k)\}$, with the same net effect.

3 Experiments

We have run some preliminary text classification experiments on the REUTERS-21578 dataset, still the most widely used benchmark in multi-label text classification research[1]. It consists of a set of 12,902 news stories, partitioned (according to the "ModApté" split we have adopted) into a training set of 9,603 documents and a test set of 3,299 documents. The documents are labelled by 118 categories; in our experiments we have restricted our attention to the 90 categories with at least one positive training example and one positive test example.

We have experimented with three different learning devices. The first is ADABOOST.MH [1], probably the best-known boosting algorithm for multi-label text classification. The second is MP-BOOST [3], a variant of ADABOOST.MH optimized for multi-label settings and which its authors have shown to obtain considerable effectiveness improvements over ADABOOST.MH. The third is an SVM-based learner as implemented in the SVM-LIGHT package [4][2], which we have fed with weighted input (in the form of standard cosine-normalized $tfidf$ weighting) so as to see how the two boosting-based systems endowed with our enhanced binary representation compare with a state-of-the-art system that makes use of full-fledged weighted input.

In all the experiments, punctuation has been removed, all letters have been converted to lowercase, numbers and stop words have been removed, and stemming has been performed by means of Porter's stemmer; word stems are thus our indexing units. Both boosting algorithms have been run with a number of iterations fixed to 1,000, while the SVM-based learner has been tested with a linear kernel and the parameters set at their default values. As a result of our encoding the number of features has increased from 20,123 to 47,087. This is a computationally non-prohibitive increase, which shows that the average number of different nonzero values that $\#(t_k, -)$ actually takes in the training set is very small (2.34 on average).

As a measure of effectiveness that combines the contributions of *precision* (π) and *recall* (ρ) we have used the well-known F_1 function, defined as $F_1 = \frac{2\pi\rho}{\pi+\rho} = \frac{2TP}{2TP+FP+FN}$, where TP, FP, and FN stand for the numbers of true positives, false positives, and false negatives, respectively. We compute both microaveraged F_1 (denoted by F_1^μ) and macroaveraged F_1 (F_1^M). F_1^μ is obtained by (i) computing the category-specific values TP_i, (ii) obtaining TP as the sum of the TP_i's (same for FP and FN), and then (iii) applying the $F_1 = \frac{2TP}{2TP+FP+FN}$

[1] `http://www.daviddlewis.com/resources/testcollections/~reuters21578/`

[2] Freely downloadable from `http://svmlight.joachims.org/`

Table 1. Results obtained on REUTERS-21578 by running several learners with different types of representation; "Binary" stands for standard presence/absence features, "Ordinal" stands for binary features obtained by encoding ordinal features into binary ones, and "Weighted" stands for cosine-normalized *tfidf*

Learner	Representation	π^μ	ρ^μ	F_1^μ	π^M	ρ^M	F_1^M
ADABOOST.MH	Binary	0.900	0.733	0.808	0.879	0.293	0.353
ADABOOST.MH	Ordinal	0.914	0.727	0.810	0.869	0.277	0.340
MP-BOOST	Binary	0.874	0.816	0.844	0.800	0.539	0.549
MP-BOOST	Ordinal	0.902	0.818	**0.858**	0.837	0.524	**0.564**
SVM-LIGHT	Weighted	0.941	0.780	0.853	0.953	0.353	0.415

formula. F_1^M is obtained by first computing the category-specific F_1 values and then averaging them across the categories $c_j \in C$.

The results of our experiments are reported in Table 1. The first observation we can make is that ordinal features encoded as binary features are usually superior to standard binary features, for both boosting devices and for both F_1^μ and F_1^M (this trend is reversed only for ADABOOST.MH with F_1^M), with MP-BOOST endowed with ordinal features being the best performer for both evaluation measures. The second observation is that, once endowed with ordinal features, MP-BOOST even outperforms SVMs with full-fledged weighted input.

4 Related Work

Our encoding of ordinal features into binary features is reminiscent of machine learning algorithms for discretizing a *continuous* (i.e., real-valued) feature t_k (see [5] for a survey and experimental comparison of less-than-recent methods, and [6,7] for two more recent surveys). These algorithms attempt to optimally subdivide the interval $[\alpha^k, \beta^k]$ on which a feature t_k ranges (where the interval $[\alpha^k, \beta^k]$ may or may not be the same for all features in the feature space) into a *partition* $I = \langle[\alpha^k = \gamma_0^k, \gamma_1^k], (\gamma_1^k, \gamma_2^k], \ldots, (\gamma_{|I|-1}^k, \gamma_{|I|}^k = \beta^k]\rangle$ of disjoint, non-necessarily equal subintervals. The partition generates a new vector (binary) representation, in which a binary feature $t_k^i \in \{0, 1\}$ indicates whether the weight of the original non-binary feature t_k belongs or does not belong to the i-th subinterval of $[\alpha^k, \beta^k]$. Our method is different from these algorithms for at least three significant reasons.

The first reason is that our features are not originally continuous, but are integer-valued; a hypothetical algorithm that operates along the lines of the ones above on integer-valued features would perform, rather than the discretization of continuous features, a sort of "segmentation of discrete features"; however, to the best of our knowledge, we are not aware of any such existing method.

The second reason is that we perform no discretization/segmentation at all, since we retain *all* the binary constraints generated from one ordinal feature in our encoding. The reason why we can do this is that boosting performs feature selection internally, choosing the best feature at each round; therefore, if some of the features we generate are non-discriminative, they will never be chosen by the boosting algorithm, and will play no role in classification decisions.

The third and probably most important reason is that, while discretization partitions a range into *disjoint* subintervals, our method actually subdivides the range R_k of an ordinal feature t_k into a sequence of chain-included subsets $S_k = \{S_k^1 \subset \ldots \subset S_k^{n(k)} = R_k\}$, since the binary features we generate check that $\#(t_k, d_j)$ is *greater than* (and not simply equal to) a given integer n. This is done in order to comply with the commonly held "monotonicity assumption" of within-document frequency [8], that states that how significant the contribution of t_k is to the semantics of d_j is a *monotonic non-decreasing* function of the number of occurrences of t_k in d_j. In fact, suppose we instead generated a binary feature of type $t_k^{(n',n'')}$ defined as $n' \leq \#(t_k, d_j) \leq n''$, with n', n'' two (possibly equal) integer numbers. A learning system would internally generate a rule saying that if $t_k^{(n',n'')} = 1$, then the evidence in favour of class c_j is α, while if $t_k^{(n',n'')} = 0$, this evidence is β. Unless $\alpha = \beta$ (in which case the feature would be useless), this would obviously violate the monotonicity assumption of within-document frequency.

References

1. Schapire, R.E., Singer, Y.: Boostexter: A boosting-based system for text categorization. Machine Learning 39(2/3), 135–168 (2000)
2. McCallum, A.K., Nigam, K.: A comparison of event models for naive Bayes text classification. In: Proceedings of the AAAI Workshop on Learning for Text Categorization, Madison, US, pp. 41–48 (1998)
3. Esuli, A., Fagni, T., Sebastiani, F.: MP-Boost: A multiple-pivot boosting algorithm and its application to text categorization. In: Crestani, F., Ferragina, P., Sanderson, M. (eds.) SPIRE 2006. LNCS, vol. 4209, pp. 1–12. Springer, Heidelberg (2006)
4. Joachims, T.: Making large-scale SVM learning practical. In: Schölkopf, B., Burges, C.J., Smola, A.J. (eds.) Advances in Kernel Methods – Support Vector Learning, pp. 169–184. The MIT Press, Cambridge (1999)
5. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: Proceeding of the 12th International Conference on Machine Learning (ICML 1995), Lake Tahoe, US, pp. 194–202 (1995)
6. Kotsiantis, S., Kanellopoulos, D.: Discretization techniques: A recent survey. GESTS International Transactions on Computer Science and Engineering 32(1), 47–58 (2006)
7. Liu, H., Hussain, F., Tan, C.L., Dash, M.: Discretization: An enabling technique. Data Mining and Knowledge Discovery 6, 393–423 (2002)
8. Zobel, J., Moffat, A.: Exploring the similarity space. SIGIR Forum 32(1), 18–34 (1998)