Text classification using string kernels

DD2434 Advanced Machine Learning Project

Daniel del Castillo
Quentin Lemaire
Jenny Stoby Höglund
Kevin Yeramian

January 2018

Abstract

The Support Vector Machine (SVM) presents itself as a powerful method for the task of text categorization. Based on the usage of a kernel function for evaluating inner products between data units in the feature space, it allows classifiers to separate high-dimensional datapoints without paying the cost of dimensionality. In 2002, Lodhi et al. introduced a novel approach to kernel functions with a special kind of kernel, called string subsequence kernel (SSK). This kernel is an inner product in the feature space generated by all subsequences of a certain length k, where a subsequence is any ordered sequence of k characters occurring in the text, though not necessarily contiguously. The main purposes of this paper are to reproduce some of the results presented in their article and to test the SSK performance against more conventional approaches. Also, a critical review of the most prominent advances in string kernels is provided. Furthermore, a computationally efficient implementation is developed by applying the approximation method designed for the SSK.

Introduction

This paper will aim to replicate the text classification method presented in the paper *Text classification using string kernels*, named String Subsequence Kernel (SSK). The classifier is a Support Vector Machine (SVM) that uses the kernel SSK. To evaluate the strength of this approach, other ways of evaluating kernels will be used for comparison. The kernels chosen are those presented in the paper *Text classification using string kernels*, namely: SVM with a Word Kernel (WK) and SVM with an n-gram kernel (NGK). Furthermore, the performance of the SVM with the three kernels mentioned before will be tested against other more conventional methods, including Naïve Bayes (NB), k-nearest neighbours (K-NN) and decision trees (DT). For clarification purposes, we must mention that along this paper we will refer to the usage of different combinations of learning algorithms and kernels, such as Decision Trees with the 3-gram kernel, for example. We are aware that Decision Trees is not a kernel-based method, so when we mention such a combination we are referring to the pre-processing that is characteristic of the n-gram kernel, i.e., the methodology that is necessary for obtaining a feature vector Finally, to attempt to improve the efficiency of the SSK it will be tested by combining the kernel with NGK and also by varying parameters such as length of the string subsequences and a weight decay factor.

State of the art in string kernels for text classification

Since the publication of [1] in 2002, a great deal of work has been directed to improve the performance of text classifiers that employ string kernels.

In [3], the author proposes encoding documents into string vectors instead of numerical vectors, which will be then fed to an SVM. This proposal is supported by the acknowledgement of two problems that are inherent to conventional methods, in which documents are encoded into numerical vectors: huge dimensionality and sparse distribution. To try to address these problems, an alternative kernel is proposed. Instead of computing the inner product between two numerical vectors to obtain a similarity measure, the notion of similarity between two string vectors is obtained by calculating the *average semantic similarity*. One possible criticism to this approach is that the average semantic similarity is based on the assumption that words that are present in the same documents tend to have similar meanings. However, this has been proved to be a rather weak link. The skip-gram model [6], for example, has shown that considering words that share the same context as similar, leads to building models that obtain better results. Nonetheless, in [3] it is shown that the string vector-based kernel function gets better performance results than the traditional numerical vector-based one in using SVM. Also, it is claimed that the proposed kernel deals better with sparse categories that contain only a few documents. Finally, the author states that by using this string kernel method, the problem of huge dimensionality is avoided. However, in the section where the string vectors were defined, it is found that dimensionality of the feature vectors were reduced by considering only the most frequent words in each document. One could argue that, depending on the training corpus, a great amount of information could be lost in this process.

In [4] the authors express their concern about high computational complexity string kernels, and thus propose a novel way of approximating SSK that is alternative to the one presented in [1]. This method is called lambda pruning. By simply dropping the lambda terms whose exponents exceed a certain threshold value, it is possible to considerably reduce the runtime and memory consumption when computing SSK. In section 7 of [1] we can observe the results of applying the approximation approach described in section 6. As an illustrative example, they show how the necessary time to generate the Gram matrix can be decreased almost five orders of magnitude, using substring length $n = 5$ and estimated document length equal to 2000. For the purpose of comparison, in section 4.2 of [4] it is observed that, for a different dataset but similar parameter values ($n = 5$ and document length 1600), the runtime can be decreased up to four orders of magnitude when executing the approximation SSK-LP. In terms of implementation, the SSK-LP is arguably simpler than the approximation method proposed in [1].

To further review the state of the art in string kernels, we shall consider now two important extensions of the string kernel proposed by Lodhi et al. in [1]. Firstly, the Word Sequence Kernel (WSK) proposed in [9] supposed the introduction of the novel idea of building string kernels from documents as word sequences. This approach addressed the main problem of SSK: the high computational complexity, due to the huge dimensionality. By considering

words instead of substrings for the feature space, the average length of the feature vectors is significantly reduced, and, in turn, a more efficient computation of the Gram matrix is possible. Although this work improved the computational efficiency and took into account word order to boost the classification performance, it still has important issues. The feature space of the word-sequence kernel is still too high dimensional and sparse, what makes the training problematic. Also, it was proved in the same work that word order had little effect on classification accuracy. Later attempts to avoid these issues crystallized in the Word-Combination Kernel (WCK), proposed in [5]. Instead of word sequences, this kernel uses word combinations, which are collections of a certain length of unique words that co-occur in the same sentence. The WCK can be defined as an inner product defined in the feature space generated by all word combinations of specified length. Unlike the WSK, this kernel does not take word order into account. This is believed to generate features that are more compatible with the flexibility of natural language. Experimental results were collected and proved that the contextual approach of the WCK yields better performance metrics than the WSK and the basic bag-of-words WK.

Methods

Conventional methods

All of the conventional methods map documents into feature vectors. To be able to properly compare documents, every entry in the feature vector is weighted with the *tf idf* weighting scheme, e.g. $\log(1 + tf) \times \log(n/df)$. [1] The weighting is important for one to know how significant each word is in the text, by looking at how many times the word is used in a document and how many times it is used overall. *tf* is the term frequency, i.e. how many times the term appears in a document, and *idf* is the inverse document frequency, which corresponds to how many documents that includes the term. Hence, words that occur many times in one document, but comparatively few times overall, are given a higher weight value. The method is used to remove so-called stop words, which are words that are most commonly used in a language.[7]

Naïve Bayes

The reason for implementing a NB model was to obtain a typical text classification baseline that would serve as reference for comparison with other models. NB classifiers are simple but surprisingly accurate and often perform very well on small datasets. This behavior might be explained by the higher rate of convergence that NB classifiers show towards their asymptotic accuracy, compared to other methods, such as logistic regression [2].

From a probabilistic perspective, NB belongs to the category of generative classifiers, which teach a model the joint probability of the inputs and the label, $p(\mathbf{x}, y)$. By applying Bayes rules to calculate the posterior probability $p(y|\mathbf{x})$, we can then pick the most likely label.

As a conventional approach for comparison with other methods, we implemented a basic multinomial NB classifier, available in the `scikit-learn` package. This is suitable for classification with discrete features, such as word counts for text classification. Our model is configured to learn the class prior probabilities and to employ Laplace additive smoothing. [2]

Decision trees

A decision tree is a classification method based on the tree representation. Each node will represent one coordinate of the input vector and each leaf of the tree will tell us the classification of our input.

The features with the biggest gain will be placed at the top of the tree. In other words, we want to place the most important features at the top to reduce the weight of the noise in the result. By doing this, it is possible to reduce the overfitting of the dataset by cutting the lowest branch which does not have an important impact on the result and is most likely to be caused by noise. This method is called pruning. [10]

k-nearest neighbours

The k-nearest classification method is the generalization of the 1-nearest neighbour, which is simply returning the same class of the nearest point in the feature space. In the case of k-nearest neighbors, the output is the class dominant between the k-nearest neighbours of the input. This method, one of the most simple machine learning methods, can also be changed to put a weight to the neighbours according to their distance with the studied point.

Therefore, the closest point will be of most importance. [10]

SVM with kernel

In the article [1] support vector machines are used with the word kernel and the n-gram kernel. Both methods are to be compared with the SSK. A support vector machine method (SVM) is a supervised learning method based on kernel functions. Data points are mapped into high dimensional feature space and then a maximal separation margin is sought. The aim is to separate each category and then be able to do prediction for future data points. It can be an infinite dimensional space depending on the kernel. [8]

Word kernel

The word kernel is a linear kernel that maps texts into feature vectors. These feature vectors can have very high dimensionality since they correspond directly to the size of the text; each entry represent the occurrence, or absence, of a word. Every entry in these feature vectors is weighted with the *tf idf* weighting scheme, which has previously been described. [1]

n-grams

n-grams is a text representation technique that converts a text document into feature vectors. Each element in the vector corresponds to how many times a string of n number of contiguous characters appear in the document. Since the technique does not interpret words, but rather a sequence of characters and spaces, it is language independent. Similarly to WK, the strings weights are calculated with the *tf idf* weighting scheme. The documents are considered similar if they share many equal subsequences. [1]

String subsequence kernel

In the paper *Text classification using string kernels* a new method is proposed based on the use of a string subsequence kernel [1]. A text document is in this method seen as a long sequence of characters that is split into subsequences of k number of characters. These subsequences of characters do not have to be contiguous. A weight λ is given to each subsequence, where larger weight is put on strings with lower proximity, i.e. a higher penalty is given to strings with large interior gaps. The classification of documents is thereafter based on the similarity between shared subsequences within the documents, i.e. the more subsequences two documents share, the more similar they are regarded.

The SSK have the following definition, cited from the original paper [1]:

Let Σ be a finite alphabet. A string is a finite sequence of characters from Σ, including the empty sequence. For strings s, t, we denote by $|s|$ the length of the string $s = s_1 \dots s_{|s|}$ and by st the string obtained by concatenating the strings s and t. The string $s[i:j]$ is the substring $s_i \dots s_j$ of s. We say that u is a subsequence of s, if there exist indices $\mathbf{i} = (i_i, \dots, i_{|u|})$, with $1 \leq i_1 < \dots < i_{|u|} \leq |s|$, such that $u_j = s_{ij}$, for $j = 1, \dots, |u|$, or $u = s[\mathbf{i}]$ for short. The length $l(\mathbf{i})$ of the subsequence in s is $i_{|u|} - i_1 + 1$. We denote by Σ^n the set of all finite strings of length n, and by Σ^* the set of all strings

$$\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n. \tag{1}$$

We now define feature spaces $F_n = R^{\Sigma^n}$. The feature mapping ϕ for a string is given by defining the u coordinate $\phi_u(s)$ for each $u \in \Sigma^n$. We define

$$\phi_u(s) = \sum_{i:u=s[\mathbf{i}]} \lambda^{l(\mathbf{i})}, \tag{2}$$

for some $\lambda \leq 1$. These features measure the number of occurrences of subsequences in the string s weighting them according to their lengths. Hence, the inner product of the feature vectors for two strings s and t give a sum over all common subsequences weighted according to their frequency of occurrence and lengths

$$\begin{aligned} K_n(s,t) &= \sum_{u \in \Sigma^n} \langle \phi_u(s) \cdot \phi_u(t) \rangle \\ &= \sum_{u \in \Sigma^n} \sum_{\mathbf{i}:u=s[\mathbf{i}]} \lambda^{l(\mathbf{i})} \sum_{\mathbf{j}:u=t[\mathbf{j}]} \lambda^{l(\mathbf{j})} \\ &= \sum_{u \in \Sigma^n} \sum_{\mathbf{i}:u=s[\mathbf{i}]} \sum_{\mathbf{j}:u=t[\mathbf{j}]} \lambda^{l(\mathbf{i})+l(\mathbf{j})}. \end{aligned}$$

Results

The documents which are to be classified are obtained from the same source, from Reuters news agency. The documents chosen were the collection Reuters-21578, which is the same collection used in

the original paper [1]. The training and test sets were obtained by different splits, and similarly to the paper the Modified Apte split was used. [1] Of this subset, only 380 articles of the 4 most frequent categories were used.

In table 1 and table 2 the calculated results are presented for the methods with of support vector machines (SVM), decision trees (DT), k-nearest neighbour with $k = 5$ (5-NN), k-nearest neighbour with $k = 3$ (3-NN), Naïve Bayes (NB) which are combined with the kernels; word kernel (WK), n-gram kernel with $n = 3$ and $n = 5$ and the string subsequence kernel approximation (SSK) also with $n = 3$ and $n = 5$ and $|\widetilde{S}| = 200$. We compute $\widetilde{S}$ as the 200 n-gram which occur most frequently in the dataset. Furthermore, different weights λ are added to the method of SVM with SSK. The results are found for four different categories, the same categories as presented in the original article; earn, acquisition, crude and corn. To be able to easily compare results with the paper [1] the mean and standard deviation were sought for F1, precision and recall.

Discussion

It is of importance to take into account the publication date of the original paper. It was published in 2002 and therefore 16 years later, the available calculation power has increased significantly. Nevertheless, the calculation time required to compute the Gram matrix for the SSK is long compared to the other methods used in this paper, even with an applied approximation method of the SSK.

The obtained SSK results in this paper differs from those presented in the original paper. The SSK results in the original paper have significantly higher accuracy than those produced in this paper. The reason for this is believed to be an error in the code but that error has not yet, despite repeatable attempts, been discovered. The length of the SSK subsequence is of importance as seen in the tables. The results are better for subsequences of length $n = 3$ than those with $n = 5$, it is thereby evident that SSK is more efficient for smaller subsequences. It is different from the conclusions in the original paper, which states the ideal length to be between $n = 4$ and $n = 7$. [1] The weight, λ, seems to only have a slight effect on the results. For the SSK with $\lambda = 0.5$ and $\lambda = 0.7$ produce fairly similar results while $\lambda = 0.1$ produce worse results in all categories except in the last category, corn.

However, the results for the methods of decision trees, k-nearest neighbours and Naïve Bayes for the WK and n-grams are overall good. Especially Naïve Bayes produces good results for medium-size datasets. These conventional methods have fairly similar results with the different types of pre-processing (kernels) we have used in Table 1 and 2. These are quite resistant methods and, therefore, it seems better to try new ones instead of changing the pre-processing. In general, SVM and NB are the two conventional methods with the best results. These methods are fast to compute and therefore allow computation for larger datasets, and not only the subset of 380 articles as in the original paper. Since the methods support faster computations, more accurate and stable results are produced.

It has not escaped our notice that the high computational complexity is, even after 16 years, the main issue of the SSK. As we have described before, there exist several ways to approximate the Gram matrix without losing much performance, but the runtime is still a challenge for single-processor computing. Even for small or moderately-sized datasets, there are simpler alternative approaches that yield better results with much less computational time and memory consumption. Because of the poor applicability of the SSK method, we shall not consider it as a general method for the task of text categorization. However, it is an interesting variant of kernel-based methods that can still be further extended and simplified, and could possibly achieve an adequate balance between complexity and computational efficiency.

Category	Kernel	Method	λ	F1		Precision		Recall	
				Mean	SD	Mean	SD	Mean	SD
earn	WK	SVM	0	0.945	0.027	0.91	0.045	0.983	0.012
		DT		0.862	0.033	0.8	0.035	0.933	0.031
		5-NN		0.894	0.026	0.821	0.036	0.983	0.024
		3-NN		0.880	0.028	0.803	0.039	0.975	0.035
		NB		0.947	0.0012	0.929	0.017	0.967	0.031
	3-gram	SVM	0	0.941	0.025	0.917	0.041	0.968	0.037
		DT		0.87	0.022	0.826	0.03	0.92	0.038
		5-NN		0.908	0.024	0.859	0.044	0.965	0.028
		3-NN		0.877	0.032	0.81	0.056	0.96	0.032
		NB		0.928	0.034	0.931	0.029	0.927	0.059
	5-gram	SVM	0	0.942	0.028	0.926	0.033	0.96	0.03
		DT		0.847	0.047	0.822	0.074	0.877	0.049
		5-NN		0.926	0.029	0.889	0.044	0.967	0.022
		3-NN		0.913	0.028	0.867	0.039	0.965	0.032
		NB		0.934	0.026	0.931	0.028	0.938	0.028
	SSK (3)	SVM	0.1	0.784	0.076	0.707	0.137	0.91	0.051
			0.5	0.835	0.045	0.778	0.081	0.908	0.035
			0.7	0.824	0.101	0.795	0.155	0.873	0.043
	SSK (5)	SVM	0.5	0.714	0.054	0.682	0.06	0.76	0.104
acq	WK	SVM	0	0.917	0.047	0.944	0.04	0.893	0.068
		DT		0.718	0.035	0.762	0.04	0.68	0.033
		5-NN		0.815	0.075	0.946	0.041	0.72	0.098
		3-NN		0.787	0.101	0.929	0.058	0.693	0.132
		NB		0.919	0.016	0.935	0.047	0.907	0.038
	3-gram	SVM	0	0.886	0.038	0.904	0.063	0.876	0.068
		DT		0.736	0.071	0.766	0.048	0.712	0.099
		5-NN		0.782	0.085	0.931	0.044	0.684	0.122
		3-NN		0.682	0.104	0.906	0.055	0.56	0.136
		NB		0.882	0.044	0.865	0.067	0.904	0.048
	5-gram	SVM	0	0.906	0.053	0.905	0.064	0.912	0.078
		DT		0.775	0.066	0.77	0.037	0.788	0.12
		5-NN		0.882	0.064	0.923	0.039	0.852	0.109
		3-NN		0.827	0.077	0.894	0.041	0.776	0.113
		NB		0.898	0.054	0.884	0.054	0.912	0.061
	SSK (3)	SVM	0.1	0.581	0.118	0.675	0.104	0.592	0.271
			0.5	0.618	0.086	0.621	0.108	0.66	0.168
			0.7	0.665	0.069	0.586	0.056	0.8	0.163
	SSK (5)	SVM	0.5	0.507	0.076	0.475	0.077	0.576	0.146

Table 1: Results for the categories, where performance is provided as the average mean of 10 runs, and the standard deviation (SD).

Category	Kernel	Method	λ	F1		Precision		Recall	
				Mean	SD	Mean	SD	Mean	SD
crude	WK	SVM	0	0.887	0.029	0.911	0.063	0.867	0.0
		DT		0.8	0.08	0.838	0.146	0.778	0.063
		5-NN		0.813	0.018	0.804	0.006	0.822	0.031
		3-NN		0.828	0.023	0.86	0.05	0.8	0.0
		NB		0.86	0.016	0.834	0.023	0.889	0.031
	3-gram	SVM	0	0.854	0.081	0.873	0.062	0.847	0.127
		DT		0.749	0.101	0.775	0.104	0.733	0.119
		5-NN		0.775	0.089	0.696	0.094	0.88	0.093
		3-NN		0.778	0.085	0.705	0.1	0.88	0.093
		NB		0.852	0.069	0.833	0.055	0.873	0.087
	5-gram	SVM	0	0.868	0.07	0.864	0.076	0.873	0.076
		DT		0.748	0.099	0.809	0.118	0.713	0.137
		5-NN		0.835	0.062	0.832	0.072	0.84	0.068
		3-NN		0.826	0.06	0.819	0.097	0.84	0.053
		NB		0.843	0.057	0.81	0.055	0.88	0.065
	SSK (3)	SVM	0.1	0.265	0.233	0.543	0.337	0.253	0.314
			0.5	0.423	0.227	0.536	0.307	0.4	0.26
			0.7	0.27	0.218	0.44	0.291	0.22	0.219
	SSK (5)	SVM	0.5	0.176	0.133	0.227	0.18	0.16	0.131
corn	WK	SVM	0	0.855	0.033	0.93	0.05	0.8	0.082
		DT		0.733	0.221	0.917	0.118	0.633	0.249
		5-NN		0.754	0.034	0.869	0.008	0.667	0.047
		3-NN		0.785	0.01	0.85	0.035	0.733	0.047
		NB		0.887	0.051	1.0	0.0	0.8	0.082
	3-gram	SVM	0	0.873	0.08	0.943	0.058	0.82	0.117
		DT		0.608	0.123	0.707	0.156	0.55	0.143
		5-NN		0.785	0.127	0.955	0.072	0.69	0.17
		3-NN		0.817	0.111	0.955	0.072	0.73	0.149
		NB		0.847	0.06	0.957	0.053	0.77	0.1
	5-gram	SVM	0	0.872	0.081	0.99	0.03	0.79	0.137
		DT		0.776	0.146	0.884	0.141	0.7	0.167
		5-NN		0.857	0.087	0.963	0.057	0.78	0.125
		3-NN		0.834	0.104	0.941	0.081	0.76	0.143
		NB		0.844	0.102	0.99	0.03	0.75	0.143
	SSK (3)	SVM	0.1	0.479	0.162	0.833	0.183	0.38	0.16
			0.5	0.295	0.251	0.481	0.423	0.23	0.21
			0.7	0.384	0.221	0.73	0.326	0.27	0.173
	SSK (5)	SVM	0.5	0.197	0.16	0.356	0.3	0.16	0.15

Table 2: Same as for Table 1.

References

[1] Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Christianini, Chris Watkins. *Text Classification using String Kernels*. University of London, 2002.

[2] Andrew Y. Ng and Michael I. Jordan. *On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes*. Advances in Neural Information Processing Systems, 14:841–848. MIT Press, 2002.

[3] Taeho Jo. *Representation of Texts into String Vectors for Text Categorization*. School of Computer and Information Engineering, Inha University, 2010.

[4] Alexander Seewald and Florian Kleedorfer. *Lambda pruning: an approximation of the string subsequence kernel*. Advances in Data Analysis and Classification, 1(3):221-239, 2007.

[5] Lujiang Zhang and Xiaohui Hu. *Word Combination Kernel for Text Classification with Support Vector Machines*. School of Automation Science and Electrical Engineering Beijing University of Aeronautics & Astronautics, 2013.

[6] Tomas Mikolov, Kai Chen, Greg Corrado and Jeffrey Dean. *Efficient Estimation of Word Representations in Vector Space*. Google Inc., 2013.

[7] Christopher Manning, Prabhakar Raghavan and Hinrich Schtze. *Introduction to Information Retrieval*. Cambridge University Press, p.117-119, 2008.

[8] Simon Tong and Daphne Koller. *Support Vector Machine Active Learning with Applications to Text Classification*. Computer Science Department Stanford University, 2001.

[9] Nicola Cancedda, Eric Gaussier, Cyril Goutte, Jean-Michel Renders. *Word-Sequence Kernels*. Journal of Machine Learning Research, Vol. 3, 2003, pp. 10591082.

[10] I. Maglogiannis, K. Karpouzis, B.A. Wallace and J. Soldatos. *Emerging Artificial Intelligence Applications in Computer Engineering*. IOS Press, 2007.