The String Subsequence Kernel in Text Classification with SVMs

Michael Giuffrida

CPSC 490 Project Proposal – 2/2/12

I. Background

The support vector machine (SVM) is a method of predicting the classification of input data. Given a set of positive and negative training examples of a class, a binary SVM classifier represents the examples in a high-dimensional space and finds the separating hyperplane between the two classes with the highest margin. If the input data is not linearly separable, it can be transformed into a higher-dimensional feature space. For efficient computation, a kernel function may be used to directly compute the inner product between data points in the feature space without explicitly representing the data in the higher-dimensional feature space, thus avoiding the high computational costs associated with feature extraction in the higher-dimensional space.

The kernel trick can be useful in text classification, with applications in bioinformatics as in [2], document classification, and spam filtering. Lodhi et al. have developed a string subsequence kernel (SSK) that compares two strings based on the number of occurrences of common substrings they contain, where each common substring is weighted based on how contiguous that substring is within the string. The feature space is indexed by all strings of *n* characters. For a particular document, the value within its feature vector corresponding to a particular *n*-string is zero if that *n*-string is not a subsequence of the document, and otherwise λ^l, where *l* is the length of the subsequence in the document and λ is a decay factor in (0, 1].

More formally, let s and t be strings from the finite alphabet Σ, where the length of $s = s_1 \ldots s_{|s|}$ is $|s|$. The substring $s_i \ldots s_j$ is denoted by $s[i{:}j]$. The *n*-string u is a subsequence of s (denoted $u = s[\mathbf{i}]$) if there exist increasing indices $\mathbf{i} = (i_1, \ldots, i_{|u|})$ such that $u_j = s_{i_j}$ for each character u_j in u. The length $l(\mathbf{i})$ of this subsequence is given by $i_{|u|} - i_1 + 1$.

[3] defines the feature mapping for a string by giving the u coordinate for each $u \in \Sigma^n$ as

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

which represents the number of occurrences of the subsequence in s, weighted by their continuities. The kernel function represents the inner product of the feature vectors for strings s and t as the sum of the products of the above u coordinate for all substrings u:

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

To compute SSK efficiently, Lodhi et al. introduce a recursive kernel:

$$K_i'(s,t) = \sum_{u\in\Sigma^i} \sum_{\mathbf{i}:u=s[\mathbf{i}]} \sum_{\mathbf{j}:u=t[\mathbf{j}]} \lambda^{|s|+|t|+i_1+j_1+2}, i = 1,..,n-1$$

so that

$$K_i'(s,t) = K_i(s,t) = 0, if \min(|s|,|t|) < i,$$

$$K_i'(sx,t) = \lambda K_i'(s,t) + \sum_{j:t_j=x} K_{i-1}'(s,t[1:j-1])\lambda^{|t|-j+2},$$

$$K_n(sx,t) = K_n(s,t) + \sum_{j:t_j=x} K_{n-1}'(s,t[1:j-1])\lambda^2$$

As Lodhi et al. show, this string subsequence kernel can be evaluated in time $O(n|s||t|)$. They explore SSK on the Reuters dataset, varying the length of the subsequence *k* as well as the decay factor λ. The F-measure of SSK is close to, but rarely better than, that of the *n*-grams kernel for this dataset, although both were better than the standard linear word kernel. The optimal sequence length was found to be fairly small (4 to 7 characters), suggesting that smaller substrings are more useful in classification than larger substrings. Furthermore, SSK was found to perform best with the decay factor λ at a very small value, thus highly penalizing interior gaps. However, the precision peaked at much higher λ values, although this did correspond to a drop in sensitivity.

The string subsequence kernel may also be useful in learning a class of strings that are labeled according to whether they contain a certain string, or one of a set of strings, as a subsequence. Given a training set, it may also be possible to partially recover the common string subsequence in question. The possibility of learning formal languages with SSK is explored in [1], whose subsequence kernel computes the inner product of two strings using the number of subsequences common to the two strings, rather than using the number of occurrences of such subsequences together with their degrees of continuity.

II. Project Focus

The primary goal of this project is to implement SSK as a kernel for an SVM, initially implementing the kernel in MATLAB. This classifier will include tuning parameters such as the weighting decay factor λ and the substring length *n*.

The SVM will then be tested on multiple data sets, including the Reuters dataset used in [3]. This will allow for a comparison with the initial results obtained by Lodhi et al. Other datasets that may be used to evaluate the behavior of SSK in training and classification include bioinformatics datasets and email corpora for spam detection. The effects of varying the parameters for SSK will then be studied and contrasted for the different datasets, with the goal of gaining a better understanding of how these parameters influence the performance of the learning algorithm.

III. Deliverables

Code

The MATLAB codebase for the SSK kernel and any other kernels implemented will be presented for evaluation, along with any preprocessing or postprocessing code developed. This code should be made publicly available.

Experiments

The results of varying the relevant parameters for various data sets will be recorded, as will the performance of any other kernels used. Hopefully, these results will help in determining how the parameters should be tuned for various classification problems.

Project Report

The final report will present the motivation for the project, describe my implementation of SSK, and present and analyze the results.

References

[1] Kontorovich, L., et al. Learning linearly separable languages. *ALT 2006*, 288-303, 2006.
[2] Leslie, C., et al. The spectrum kernel: A string kernel for SVM protein classification. *Pacific Symposium on Biocomputing*, 7:566-575, 2002.
[3] Lodhi, H., et al. Text Classification using String Kernels. *Journal of Machine Learning Research*, 2:419-444, 2002.