Kernel Methods for Pattern Analysis

Project Report

Konstantin Tretyakov, Jelena Zaitseva

24. november 2004

General Notes

The assigments of the project were implemented in SciLab[1], as it was the closest freeware alternative to Matlab. We also developed a small standalone C++ application for cross-language retrieval just for the fun of it (and also in order to have an excuse for doing the project together). Code for kernel calculations is also written in C++.

We used the English and German versions of the Finnish constitution for the cross-language retrieval task, and the English version for the multiclass classification task.

The report is accompanied by lots of files organized in four directories.

- The `kernelmethods/` directory contains the SciLab sources for all the algorithms (apart from kernel computations).
- The `crosslang/` directory contains the C++ source code of the standalone application for cross-language retrieval. In particular, code for kernel computations is there.
- Directory `reports/` contains algorithm precision summaries and images of kernels.
- `data/` contains the data used.

Task 1: Evaluating the 3-mer kernel

For the subroutine to evaluate the k-mer kernel on a set of strings see function `KMerKernel::getKernelMatrix` in `crosslang/src/Kernels.cpp`.

The subroutine implicitly traverses a trie structure of all k-mers and for each node of the trie tracks all the positions in all the strings, where a k-mer, corresponding to that node ends. Such an approach allows to calculate the whole kernel matrix in approximately $O(A^k n + ln)$ time, where A is the alphabet size,

[1] http://www.scilab.org

n is the number of strings and l the length of each string. For $k = 3$ and $n = 262$ the speedup over a naive $O(lkn^2)$ implementation is substantial.

The plot of the resulting kernels are in files `K_3mer.gif` and `K_bow.gif` in the `reports/` directory. There are also plots of the normalized and centered versions of the kernels in `K_3mer_nc.gif` and `K_bow_nc.gif`. One may note the block structure in all cases.

We tried using the kernels with all combinations of normalization and centering, and results indicated nearly no difference in resulting precision. However, when both centering and normalization was used, the precision of cross-language retrieval was good for wider range of algorithm parameters
(see `reports/K_3mer_cca_results.txt`).

Task 2: CCA Cross-Language Retrieval

We chose he range of values for the number of components k as 5, 10, 20, 50, 80. The range of values for the regularization parameter r was 0.001, 0.01, 0.1, 1, 10.

In case of a normalized and centered 3-mer kernel the best cross-validation precision was attained at $k = 20$, $r = 0.01$. The corresponding test precision was 0.96 (i.e. the test error was $1 - 0.96 = 0.4 = 4\%$). Of course, the results depend a bit on the way the random split into test/training data was made.

Detailed reports for both the BoW and 3-mer kernels with all combinations of centering and normalization are in files `reports/K_3mer_cca_results.txt` and `reports/K_bow_cca_results.txt`.

The code for the kRCCA algorithm is in the `kernel_rcca` function in the file `kernelmethods/kernel_crosslang.sci`. The code for the whole task (the cross-validation, basically) is available in the
`kernelmethods/cross_language_retrieval.sce` file. This file may be executed in SciLab to obtain a report similar to those mentioned before.

Task 3: Kernel Adatron for Multiclass Classification

The table showing the dependence of the cross-validation precision on the choice of the regularization parameter C is presented in the files
`reports/K_3mer_multiclass_results.txt` and
`reports/K_bow_multiclass_results.txt` for 3-mer and BoW kernels correspondingly.

In case of a BoW kernel (normalized, not centered), the optimal C was determined by cross-validation as 10 or 100 for AvA, and 0.001, 0.01, 0.1 or 1 for OvA. (The range of C values tested was 0.0001, 0.001, 0.01, 0.1, 1, 10, 100).

In case of AvA, the value of the regularization parameter found by cross-validation maximised also the test precision, which was equal to 0.77.

In case of OvA, the test precision corresponding to found C was 0.72, whereas for $C = 10$ or $C = 100$ it was higher: 0.82.

The code for the kernel adatron algorithm is in the `kernel_adatron` function in the file `kernelmethods/kernel_multiclass.sci`. The script that was used to do the cross-validation and produce the reports is in the file `kernelmethods/multiclass_classification.sce`.