Data Challenge of Kernel Methods for ML

Angelo Ortiz

10th April 2022

Introduction

In the context of the module 'Kernel Methods for Machine Learning', I faced a data challenge[1] that consisted in classifying some preprocessed images (likely from CIFAR-10.) The goal was to implement at least one kernel method throughout the learning pipeline. I was provided with 5000 labelled images and 2000 images whose label was to be predicted. The images are 32-pixel long and 32-pixel wide.

My plan was to, first, reproduce in PyTorch the results from the paper [1], which learns feature maps for each image in an unsupervised, kernel-based manner, which are then fed to a linear, multi-class SVM solver; second, I would learn the kernel features in a fully supervised way [2].

1 Convolutional Kernel Networks

Following [1], the goal is to use the kernel trick to lift the space of pixel locations into some reproducing kernel Hilbert space (RKHS), so that the embeddings of the images (feature maps) in the RKHS are more easily classified, notably via a linear SVC. The reproducing kernel is based on the Gaussian kernel, so for the sake of computability, the kernel is approximated. This approximation can be seen as convolution applied to a non-linearity (first link to CNNs). The kernel approach is multi-layered, so that one can learn feature maps of increasingly larger image patches so that the final feature maps are representative of the image and invariant to shifts, akin to CNNs (second link).

I used [3] for the 'soft-binning' of gradient orientations for input maps corresponding to gradient maps. Moreover, for the initialisation of the matrix W I used k-means with the k-means++ initialisation scheme [4].

2 SVM solver

Since the number of features in the output maps is large, I decided for a fast linear SVM solver [5].

3 Results & Remarks

My official results are 0.105 on the public leader-board and 0.09 on the private one. Nevertheless, they correspond to a random class assignment since I was unable to make my method work in time.

At the time of writing, my method seems to be working better (0.205 in both leaderboards), although the results are far from the top ones. My *best* architecture[2] was based on patch-map input maps with a number of output filters fixed at 100 and 600 (training took 2.3118e+03 s, while the computation of the output maps took 3.7477e+01 s, so clearly the method needs improving.)

[1] The code is available at `https://github.com/angelo-ortiz/Kernels-Competition`.
[2] To run the experiments, please run `start.py` and `start_svm.py`, in that order.

References

[1] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid, "Convolutional kernel networks," 2014.

[2] J. Mairal, "End-to-end kernel learning with supervised convolutional kernel networks," 2016.

[3] M. Paulin, M. Douze, Z. Harchaoui, J. Mairal, F. Perronnin, and C. Schmid, "Local Convolutional Features with Unsupervised Training for Image Retrieval," in *ICCV - IEEE International Conference on Computer Vision*, (Santiago, Chile), pp. 91–99, IEEE, Dec. 2015.

[4] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, (USA), p. 1027–1035, Society for Industrial and Applied Mathematics, 2007.

[5] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," Tech. Rep. MSR-TR-98-14, Microsoft, April 1998.