Supplementary Material for "Visual Recognition by Counting Instances: A Multi-Instance Cardinality Potential Kernel"

Hossein Hajimirsadeghi Wang Yan Arash Vahdat Greg Mori
School of Computing Science, Simon Fraser University, Canada
hosseinh@sfu.ca, wyan@sfu.ca, avahdat@sfu.ca, mori@cs.sfu.ca

1. Computational Complexity of the Proposed Method

First, we analyze the time complexity of computing the Cardinality Kernel:

$$\tilde{k}(\mathbf{X}_p, \mathbf{X}_q) = \sum_{i=1}^{m_p} \sum_{j=1}^{m_q} k_x(\mathbf{x}_{pi}, \mathbf{x}_{qj}) P(y_{pi} = 1|\mathbf{X}_p) P(y_{qj} = 1|\mathbf{X}_q). \quad (1)$$

Assume the evaluation of the primitive kernel k_x takes $O(d)$ time, where d is the size of the instance feature vectors. Consequently, $k_x(\cdot,\cdot)$ between all instance pairs of two bags X_p and X_q can be computed in $O(m_p m_q d)$ time. As we explained in Sec. 3.2, the time complexity of computing the marginal probabilities $P(y_i|Y, \mathbf{X})$ is $O\left(m \log^2 m\right)$. Thus, the kernel in (1) can be evaluated in $O(m_p m_q d + m_p \log^2 m_p + m_q \log^2 m_q)$ time. As a result, the computational complexity of prediction with this kernel in a standard SVM for a single bag $\mathbf{X}_p$ is $O(N_{sv}\, \bar{m}\, m_p\, d + N_{sv}\, \bar{m}\, \log^2 \bar{m} + m_p\, \log^2 m_p)$, where N_{sv} is the number of support vectors and $\bar{m}$ is the maximum number of instances in the training bags.

Now, we analyze the computational complexity of training the Cardinality Kernel. First, the parameters of the Cardinality Model should be learned. Learning this HCRF with regularized likelihood maximization takes $O(N_{iter}\, N\, \bar{m} \log^2 \bar{m} + N_{iter}\, N\, \bar{m}\, d)$ time, where N is the number of training bags and N_{iter} is the number of iterations of the gradient ascent algorithm. The kernel matrix can be computed in $O(N^2\, \bar{m}^2\, d + N\, \bar{m}\, \log^2 \bar{m})$ time. Finally, assuming the quadratic programming to solve the SVM dual takes $O(N^3)$ time[1], the computational complexity of the entire algorithm is $O(N_{iter}\, N\, \bar{m} \log^2 \bar{m} + N_{iter}\, N\, \bar{m}\, d + N^2\, \bar{m}^2\, d + N^3)$.

[1] In our experiments, we used the LIBSVM [5] solver, which can be much more efficient than $O(N^3)$ in practice.

We performed our experiments on an Intel(R) Core(TM) i7-2600 CPU @ 3.40GHz. As a numerical example, in the collective activity recognition dataset, used in the experiments in Sec. 4.1 (which consists of 1908 training bags and 639 test bags with average 5 instances per bag and the instances are 240 dimensional), the total training time was around 10 minutes versus 7 minutes for test time.

2. Parameter setting guidelines for the proposed Cardianlity Kernel method

To train the cardinality model, two hyper-parameters should be set: the regularization weight for likelihood optimization of the Cardinality Model (i.e., λ), and the regularization weight for training SVM (i.e., C). In our experiments we used the standard grid search over a set of predetermined values of λ and C (e.g., powers of 10) to set these parameters. However, this requires a quadratic number of runs of the algorithm, which might be inefficient. As a faster alternative, we found that just running MI-Kernel is effectively enough to estimate the value of parameter C. Next, we fix this parameter and run the Cardinality Kernel method to find the best estimate for λ.

Another setting is to initialize the learning parameters of the Cardinality Model (i.e., $\boldsymbol{\theta}$) for regularized likelihood maximization. Actually, this is a non-convex optimization and sensitive to initialization. In all our experiments we initialized the parameters $\boldsymbol{\theta}$ to zero. In fact, since the resulting kernel is finally plugged into SVM, and SVM relies on a convex optimization, the whole algorithm is fairly robust to initialization. For example, even if we freeze $\boldsymbol{\theta} = \mathbf{0}$ and do not train the cardinality model, it can be shown the resulting kernel is equivalent to MI-Kernel, which is an effective multi-instance algorithm [7].

978-1-4673-6964-0/15/$31.00 ©2015 IEEE

3. Additional Experiments on Collective Activity Recognition Dataset

For the collective activity dataset the standard evaluation protocol, introduced by Choi et al. [6], is multi-class classification by considering all the 5-class confusion matrix. However, Amer et al. [2, 4, 3, 1] use a different evaluation setting, training binary classifiers for each class and computing the accuracy of detection. Finally, the average accuracy over all the classes is reported. In this way, significantly higher accuracies are obtained. Table 1 shows the comparison with the relevant methods in [2, 4, 3, 1], using this evaluation setting. Note that it is not a completely fair comparison, since the training/test splits might be different[2].

Table 1. Comparing the proposed Cardinality Kernel method with the relevant methods in [2, 4, 3, 1] based on binary classification accuracy on the collective activity dataset.

Class	Ours	[2]	AOG [4]	HiRF [1]	HiRFnt [1]	ST-AOG[3]
Cross	86.1	69.9	77.2	76.8	81.2	81.1
Wait	84.4	74.1	78.3	74.3	78.4	83.9
Queue	95.6	96.8	95.4	81.1	96.2	97.5
Walk	86.7	72.2	74.7	84.1	77.3	83.4
Talk	99.8	99.8	98.4	99.3	99.6	98.8
Avg	90.5	82.5	84.8	83.1	86.6	88.9

4. Additional Experiments on TRECVID MED11 dataset, Using Dense SIFT featuers

In our second experiment on TRECVID MED11 dataset, we used dense SIFT features quantized into 1500 codewords to have a fair comparison with the experiments in [8]. However, better results can be achieved by using more codewords. Table 2 shows the results with 1,500 and 20,000 codewords.

Table 2. The results of our proposed Cardinality Kernel method on TRECVID MED11 dataset, using dense SIFT features quantized into 1,500 and 20,000 codewords.

Event	Cardinality Kernel (1,500 codewords)	Cardinality Kernel (20,000 codewords)
6	2.8 %	**8.3** %
7	5.8 %	**21.0** %
8	17.0 %	**38.9** %
9	8.8 %	**13.7** %
10	1.3 %	**3.9** %
11	**3.4** %	3.3 %
12	10.7 %	**11.5** %
13	4.7 %	**11.5** %
14	4.9 %	**12.6** %
15	1.4 %	**4.2** %
mAP	6.1 %	**12.9** %

References

[1] M. R. Amer, P. Lei, and S. Todorovic. Hirf: Hierarchical random field for collective activity recognition in videos. In *European Conference on Computer Vision (ECCV)*, pages 572–585. Springer, 2014. 2

[2] M. R. Amer and S. Todorovic. A chains model for localizing participants of group activities in videos. In *International Conference on Computer Vision*, 2011. 2

[3] M. R. Amer, S. Todorovic, A. Fern, and S.-C. Zhu. Monte carlo tree search for scheduling activity recognition. In *International Conference on Computer Vision*, 2013. 2

[4] M. R. Amer, D. Xie, M. Zhao, S. Todorovic, and S. C. Zhu. Cost-sensitive top-down/bottom-up inference for multiscale activity recognition. In *European Conference on Computer Vision (ECCV)*, 2012. 2

[5] C. Chang and C. Lin. Libsvm: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2(3):27, 2011. 1

[6] W. Choi, K. Shahid, and S. Savarese. What are they doing?: Collective activity classification using spatio-temporal relationship among people. In *9th International Workshop on Visual Surveillance*, 2009. 2

[7] T. Gärtner, P. Flach, A. Kowalczyk, and A. Smola. Multi-instance kernels. In *International Conference on Machine Learning (ICML)*, pages 179–186, 2002. 1

[8] K.-T. Lai, F. X. Yu, M.-S. Chen, and S.-F. Chang. Video event detection by inferring temporal instance labels. In *Computer Vision and Pattern Recognition (CVPR)*, 2014. 2

[9] T. Lan, Y. Wang, W. Yang, S. N. Robinovitch, and G. Mori. Discriminative latent models for recognizing contextual group activities. *IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI)*, 34(8):1549–1562, 2012. 2

[2] All the methods use almost 1/3 of the video clips for test and the rest for training. However, the exact splits are not given in [2, 4, 3, 1]. In our experiments we used the same splitting as in [9], i.e., the video clips 7, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 26, 27, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44 for training and video clips 1, 2, 3, 4, 5, 6, 8, 9, 10 ,11, 25, 28, 29 for test