A. Additional Experiments on DisMEC

DisMEC (Babbar & Schölkopf, 2017) is a distributed extreme multi-label learning framework based on one-versus-rest linear classifiers with explicit model size controlled by pruning small weights. Unlike other methods we have compared in Section 5, DisMEC mainly focuses on parallelizing an extremely large number of one-versus-all classifiers in large-scale distributed settings with double layers of parallelization (multi-core and multi-machine).

DisMEC's primarily advantage is that it does not make any low-rank or sparsity assumption for the data and thus prediction performance is better, and its model size is reasonably small due to weight pruning. However, it has the same time complexity as the naive one-versus-all method and requires much more computation than all other methods we have compared in Section 5. For example, on dataset Wiki10-31K, our algorithm needs less than 20 minutes on 1 core (refer to Table 1), while DisMEC requires 10 minutes on 300 cores as reported in (Babbar & Schölkopf, 2017) and we record about 450 minutes training time using 4 cores.

Table 5: Experiments on DISMEC. Time refers to prediction times in seconds. Size is the modelsize in megabytes.

	DISMEC				GBDT-SPARSE (proposed)			
	Time	Size	*P@1*	*P@3*	Time	Size	*P@1*	*P@3*
Mediamill	0.59	0.087	87.77	70.25	0.60	3.54	84.23	67.85
Delicious	0.24	3.4	66.80	61.79	0.13	4.76	69.29	63.62
Wiki10-31K	771.8	880	84.12	74.71	1.30	85.81	84.34	70.82

We use the DisMEC implementation from its authors [4]. We found that in their experiment implementation, a TF-IDF (term frequency inverse document frequency) feature conversion is used. Using TF-IDF features can improve prediction accuracy for text based datasets, and we found that it is necessary to use TF-IDF to get a good accuracy for Wiki10-31K. Therefore, we pre-process Wiki10-31K using TF-IDF for DisMEC in this section (we do not use TF-IDF pre-processing in all other experiments). Due to our limited computing resources, we only include Mediamill, Wiki10-31K and Delicious in this experiment. The result is shown in Table 5. DisMEC achieves similar performance with our method, but note that DisMEC requires much more computation resources than our method. Larger Datasets like Delicious-200K is practically unfeasible on a single machine (with only a few cores) using DisMEC.

B. Prediction time for GBDT-Sparse

In many real world applications, only top-B labels are needed with very small B (typically 1,3,5). In those cases, we can further reduce the prediction time to $O(Tk\log B)$. To do that, we need a hash (with $O(Tk)$ size) and a min-heap Q. The algorithm scans through all the elements in the prediction vectors (each contains k $(idx, value)$ pairs) for each tree $h_1(\boldsymbol{x}_i), \cdots, h_m(\boldsymbol{x}_i)$. For each $(idx, value)$ pair, we first use hash to add the value to the corresponding index p_{idx}. If the index is already in the heap, then update the corresponding value. If $Q.size()$ is smaller than B, then add the (idx, p_{idx}) pair to Q. Otherwise compare p_{idx} with $Q.min()$, and replace the minimum number in Q by p_{idx} if p_{idx} is larger than $Q.min()$. Since the size of Q is always $\leq B$, the complexity is $O(Tk\log B)$.

[4]https://sites.google.com/site/rohitbabbar/code/dismec