Learning to Calibrate and Rerank Multi-label Predictions (Supplementary Material)

Cheng Li, Virgil Pavlu, Javed Aslam, Bingyu Wang, and Kechen Qin

Khoury College of Computer Sciences, Northeastern University, Boston, USA
{chengli, vip, jaa, rainicy}@ccs.neu.edu, qin.ke@husky.neu.edu

A BR-rerank vs. CBM in Calibration

In Table 4 of the main paper, we showed that BR-rerank and CBM have similar classification performance. But BR-rerank has the advantage over CBM in confidence calibration. Even though CBM is designed to directly estimate the joint label probabilities [2], its estimations are often over-confident on new test data. Figure 1 below compares BR, BR-rerank and CBM in terms of prediction confidence calibration on the MSCOCO dataset. As we can see from the plots, BR prediction scores are under-confident, BR-rerank prediction scores are well-calibrated, and CBM prediction scores are over-confident.

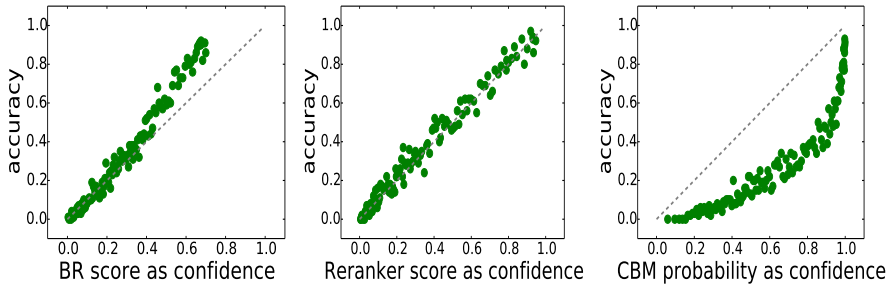

Figure 1: Comparison between BR, BR-rerank and CBM in terms of confidence calibration on MSCOCO test dataset. Each dot represents a group of 100 set predictions with similar confidence scores. The average confidence score in the group is used as x-coordinate, and the average prediction accuracy is used as y-coordinate.

B Illustration of BR-rerank Prediction Details

In Figure 2 of the main paper, we have given two example images from the MSCOCO datasets on which BR-rerank corrects BR's predictions. In Table 3 of the main paper, we have shown how BR-rerank scores and ranks prediction candidates differently than the original BR. The detailed computation that leads to the reranker scores were not provided in the main paper due to space constraints. Here in Figure 2 below we provide the prediction computation details for the first example image, including the BR marginal probabilities and the features extracted from the candidate sets.

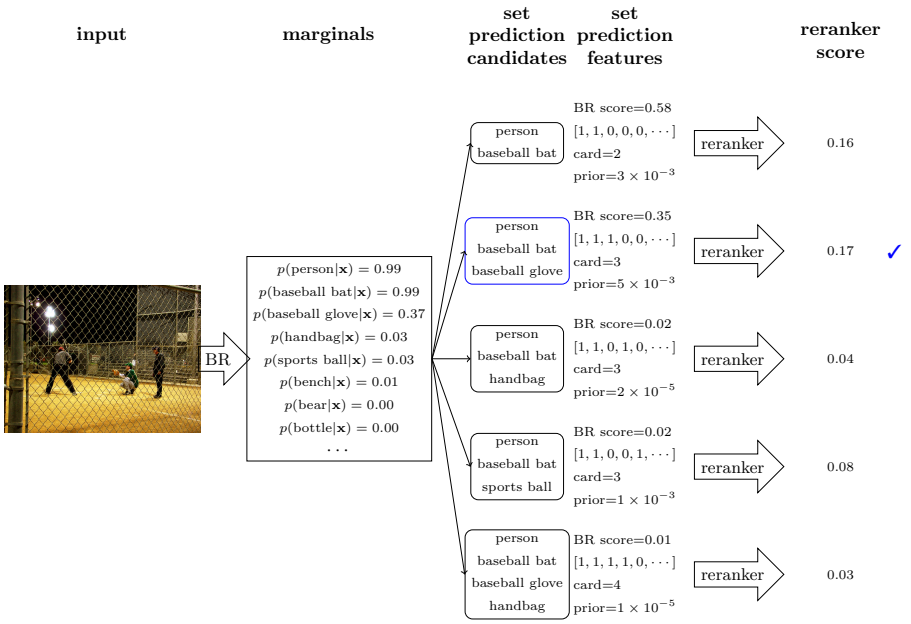

Figure 2: BR-rerank prediction details for the input test image. The "marginal" column shows the individual label probabilities estimated by BR. Note that the label `baseball glove` has a probability below the 0.5 threshold, and therefore will not be included in BR's predictions. The "set prediction candidates" column shows the top-5 set prediction candidates with the highest BR scores generated by dynamic programming based on BR marginals. The "set prediction features" column shows, for each set candidate, its BR score, its binary encoding, its cardinality and its prior probability. The "reranker score" column shows the calibrated BR-rerank confidence score for each set prediction candidate. For this image, BR predicts the incorrect set {`person,baseball bat`} with confidence 0.58. BR-rerank predicts the correct set {`person, baseball bat, baseball glove`} with confidence 0.17.

C The Impact of Number of Candidates in Reranking

Figure 3 shows BR-rerank classification set accuracy as a function of number of candidates K. When only the top-1 candidate is considered, BR-rerank is the same as BR. Significant improvement can be achieve by simply considering and reranking the top-10 candidates. Further increasing K does not give consistent improvement and sometimes causes the performance to drop slightly.

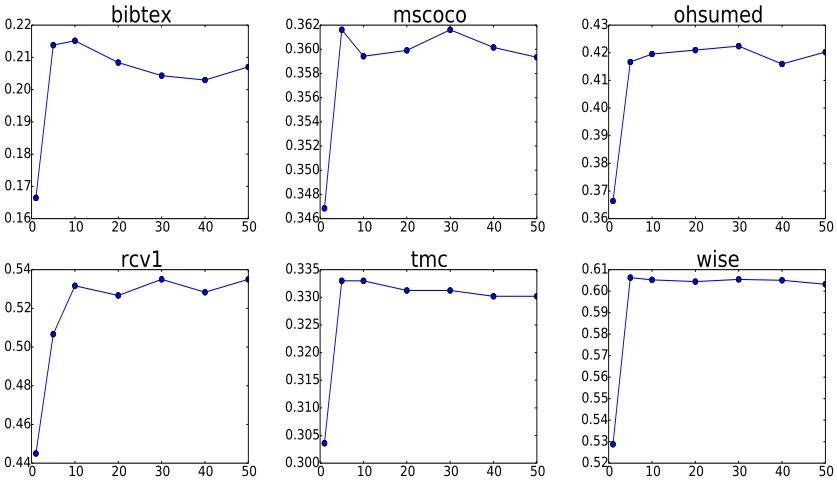

Figure 3: BR-rerank classification set accuracy as a function of the number of candidates K obtained from BR. Each subfigure is for a different dataset. X-axis is the number of candidates K, Y-axis is the set accuracy on test data.

D An Algorithm for Generating Top-K Set Prediction Candidates from BR

The BR-rerank classifier reranks the top-K set prediction candidates generated from BR. An efficient dynamic programming algorithm for generating the required candidates is described in [2]. It works as follows:

Algorithm 1 Generating the K-best prediction candidates from BR

Input: instance x and a BR classifier
Compute individual label probabilities based on BR: $p_l = p(y_l = 1|x), l = 1, 2, ..., L$
Initialize an empty priority queue Q^k, and empty list C and an empty label set $\mathbf{y}_{best}$
for $\ell = 1, 2, ..., L$ **do**
  if $p_l > 0.5$ **then**
    add l to $\mathbf{y}_{best}$
  end if
end for
$Q^k.enqueue(\mathbf{y}_{best})$
while $|C| < K$ **do**
  $\mathbf{y} = Q^k.dequeue()$
  add $\mathbf{y}$ to C
  for $\ell = 1, 2, ..., L$ **do**
    Generate $\mathbf{y}'$ by flipping the ℓ-th bit of $\mathbf{y}$
    if $\mathbf{y}'$ has not been added to Q before **then**
      $Q^k.enqueue(\mathbf{y}')$
    end if
  end for
end while
Output: C

E Hyper Parameters Tuning in Experiments

- BR: elastic-net regularization penalty strength $\lambda \in \{0.0001, 0.000001\}$; L1 ratio $\in \{0.1, 0.5\}$; training iteration is decided by monitoring the validation performance after each iteration.
- GB calibrator: shrinkage = 0.1; regression tree weak learner has 10 leaves; the minimum number of instances per leaf is 5; training iteration is tuned on validation set.
- BR-rerank: apart from setting the hyper parameters mentioned in GB calibrator, we also set the number of candidates $K = 10$.
- 2BR and DBR: we use the validation data to decide whether the stage-2 model should take instance features x as par of the input; for the GB base learner, we tune the number of training iterations using validation data; we set shrinkage = 0.1; regression tree weak learner has 10 leaves; the minimum number of instances per leaf is 5.
- Probabilistic Classifier Chains (PCC): following the common practice, labels are arranged by decreasing frequency; for the GB base learner, we tune the number of training iterations using validation data; we set shrinkage = 0.1; regression tree weak learner has 10 leaves; the minimum number of instances per leaf is 5; for prediction, we use beam search with width 5.
- Random k-label-sets (RAKEL): The number of labels used in each labelset $K \in \{2, 4, 6, \ldots, 30\}$.
- Multi-label k Nearest Neighbors (KNN): number of neighbors $k \in \{1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21\}$; the smoothing parameter $s \in \{0.5, 0.7, 1.0\}$.
- Deep Value Network (DVN): whether or not to includes a linear layer after the two-layer perceptron.
- Predict and Constrain (PC): number of hidden units for the linear part $h \in \{100, 150, 200\}$.
- PD-Sparse (PDS): L1 regularization weight, $\lambda \in \{1, 0.1, 0.01, 0.001, 0.0001\}$; in order to predict a subset of labels as opposed to simply ranking labels, we tune the score threshold.
- SPEN: we use the hidden layer sizes mentioned in the original paper.

F Experiments with Monotonicity and Another GB Variant

We conducted additional experiments on imposing partial monotonicity in GB. Since the GB score is the sum of regression tree scores, it suffices to impose monotonicity constraints in each tree. We follow the method implemented in the xgboost package [1] which works as follows: We associate a lower bound and an upper bound with each tree node (including intermediate nodes). As the tree grows, each node first inherits its parent's bounds and then tightens the bounds as new constraints are introduced. A leaf always outputs a value within its bounds. When a node is split into two children by a monotonic feature, assuming data with smaller feature value goes left, we compute the middle point between the output of the left child and the output of the right child (they are treated as leaves), and set the upper bound of the left child and the lower bound of the right child to be this middle point. When a node is split with a non-monotonic feature, its children just inherit its bounds without further tightening.

We also tested anther variant of GB, named GB-KL, which has a sigmoid transformation on top of the ensemble score and is trained by minimizing KL divergence. We compare it with GB-MSE, which does not employ the sigmoid transformation and is trained by minimizing square error.

We tested all 4 configurations of calibrator and monotonicity and the results are shown in Tables 1,2,3,4. All 4 configurations have similar calibration and prediction performance.

calibrator	monotonicity	BIBTEX	OHSUMED	RCV1	TMC	WISE	MSCOCO
GB-MSE	no	0.0682	0.1889	0.1229	0.1800	0.1472	0.1434
GB-MSE	yes	0.0653	0.1854	0.1227	0.1783	0.1468	0.1438
GB-KL	no	0.0696	0.1904	0.1260	0.1794	0.1476	0.1437
GB-KL	yes	0.0666	0.1850	0.1253	0.1772	0.1467	0.1436

Table 1: BR prediction calibration performance in terms of MSE (the smaller the better).

calibrator	monotonicity	BIBTEX	OHSUMED	RCV1	TMC	WISE	MSCOCO
GB-MSE	no	0.0719	0.0467	0.1262	0.0319	0.1022	0.0825
GB-MSE	yes	0.0783	0.0475	0.1366	0.0347	0.1029	0.0823
GB-KL	no	0.0753	0.0475	0.1236	0.0330	0.1014	0.0824
GB-KL	yes	0.0782	0.0496	0.1299	0.0351	0.1028	0.0824

Table 2: BR prediction calibration performance in terms of sharpness (the bigger the better).

calibrator	monotonicity	BIBTEX	OHSUMED	RCV1	TMC	WISE	MSCOCO
GB-MSE	no	21.5	42.0	53.2	33.3	60.5	35.9
GB-MSE	yes	22.1	41.9	52.7	32.9	60.4	36.0
GB-KL	no	21.1	41.1	51.8	33.0	60.3	36.2
GB-KL	yes	22.1	41.4	52.2	33.1	60.5	36.1

Table 3: Prediction performance of BR-rerank in terms of set accuracy.

calibrator	monotonicity	BIBTEX	OHSUMED	RCV1	TMC	WISE	MSCOCO
GB-MSE	no	42.2	67.5	78.8	66.8	75.4	73.2
GB-MSE	yes	43.1	67.2	78.6	66.6	75.5	73.4
GB-KL	no	41.8	66.8	78.2	66.4	75.1	73.3
GB-KL	yes	42.4	67.1	78.2	66.5	75.3	73.3

Table 4: Prediction performance of BR-rerank in terms of instance F1.

G Implementations Used in Experiments

Table 5 shows the code we ran in our experiments.

Method	Implementation
BR, BR-rerank, 2BR, DBR, PCC, CRF, CBM	ours: `https://github.com/cheng-li/pyramid`
RAKEL, KNN	`https://github.com/scikit-multilearn/scikit-multilearn`
DVN	`https://github.com/gyglim/dvn`
PC	`https://github.com/Natalybr/predict_and_constrain`
PDS	`http://www.cs.utexas.edu/~xrhuang/PDSparse/`
SPEN	`https://github.com/davidBelanger/SPEN`

Table 5: Implementations used in our experiments.

References

[1] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.

[2] Cheng Li, Bingyu Wang, Virgil Pavlu, and Javed A. Aslam. Conditional bernoulli mixtures for multi-label classification. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 2482–2491, 2016.