Deep Learning with a Rethinking Structure for Multi-label Classification

Yao-Yuan Yang, Yi-An Lin, Hong-Min Chu, Hsuan-Tien Lin

Department of Computer Science & Information Engineering, National Taiwan University

Introduction

Problem definition

- train set $\mathcal{D} = \{(\mathbf{x}^{(n)}, \mathbf{y}^{(n)})\}_{n=1}^{N}$
- label vector $\mathbf{y} \in \{0, 1\}^K$
  - $\mathbf{y}[k] = 1$ if and only if the k-th label is relevant to $\mathbf{x}$
- learn a classifier f that maps $\mathbf{x}$ to $\mathbf{y}$
- test data $(\mathbf{x}, \mathbf{y})$, prediction $\hat{\mathbf{y}} = f(\mathbf{x})$
- goal is to make $\hat{\mathbf{y}}$ close to ground truth $\mathbf{y}$

Evaluation

- cost function $C(\mathbf{y}, \hat{\mathbf{y}})$: the cost of predicting $\mathbf{y}$ as $\hat{\mathbf{y}}$
- common cost functions: Hamming loss, Rank loss, F1 score, Accuracy score

Key aspects for Multi-label classificaion (MLC)

- Label correlation: use the existence of other labels to make better prediction
- Cost information: perform better when the evaluation criterion is known

MLC Baselines

	Binary Relevance (BR)	Classifier Chain (CC)
dog	$f_1(\mathbf{x})$	$\hat{y}[1] = f_1(\mathbf{x})$
rabbit	$f_2(\mathbf{x})$	$\hat{y}[2] = f_2([\mathbf{x}; \hat{y}[1]])$
cat	$f_3(\mathbf{x})$	$\hat{y}[3] = f_3([\mathbf{x}; \hat{y}[1]; \hat{y}[2]])$
guinea pig	$f_4(\mathbf{x})$	$\hat{y}[4] = f_4([\mathbf{x}; \hat{y}[1]; \hat{y}[2]; \hat{y}[3]])$
shark	$f_5(\mathbf{x})$	$\hat{y}[5] = f_5([\mathbf{x}; \hat{y}[1]; \hat{y}[2]; \hat{y}[3]; \hat{y}[4]])$

- BR learns each label independently and does not consider label correlation
- CC considers label correlation by predicting labels sequentially
  - can suffer from label ordering issue
  - can be seen as forming memory between labels

Recurrent Neural Network (RNN)

- architecture designed to solve sequence prediction problem
- $\mathbf{x}^{(i)}$: i-th iteration feature vector
- RNN learns two transformations
  - feature transformation $\mathbf{U}(\cdot)$
  - memory transformation $\mathbf{W}(\cdot)$

With a sequence of inputs $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(B)}\}$, the RNN will output $\{\mathbf{o}^{(1)}, \dots, \mathbf{o}^{(B)}\}$.

$$\mathbf{o}^{(1)} = \sigma(U(\mathbf{x}^{(1)}))$$
$$\mathbf{o}^{(i)} = \sigma(U(\mathbf{x}^{(i)}) + W(\mathbf{o}^{(i-1)}))$$

RethinkNet

- utilize RNN for sequence prediction to model label correlation
- treating MLC problems as a sequence (predict a sequence of label vectors instead of sequence of labels like CC)
- use the label vector from previous prediction to fine-tuned the next prediction

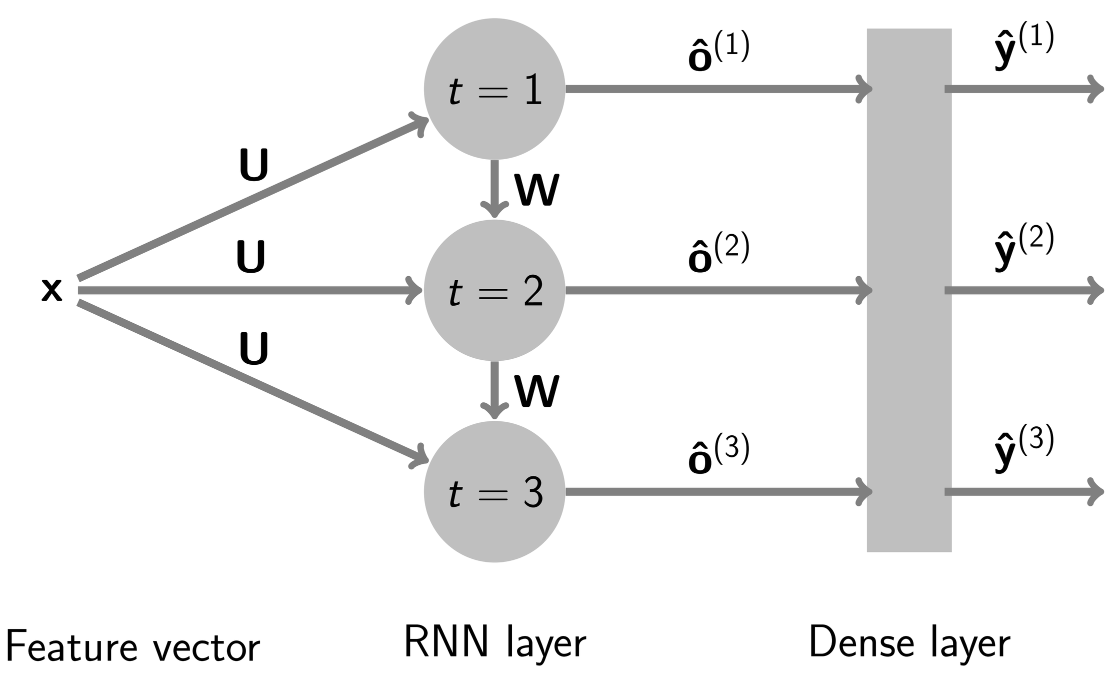
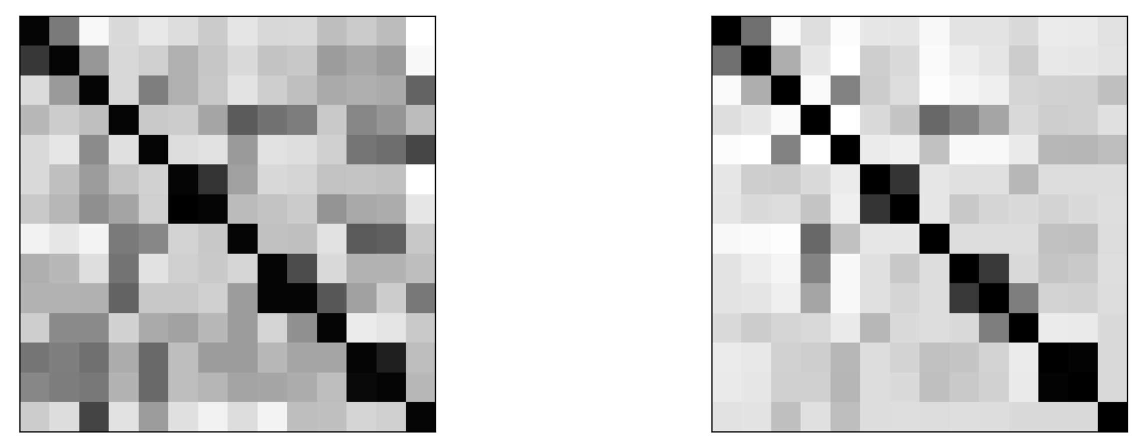


Figure: The architecture of the proposed RethinkNet model with 3 rethink iterations. The Dense layer just represents an arbitrary architecture for further feature extraction. Label vector $\hat{\mathbf{y}}^{(3)}$ is the final output.

- when $t = 1$, prediction $\hat{\mathbf{o}}^{(1)}$ is basically BR (not considering label correlation)
- when $t > 1$, previous prediction $\mathbf{o}^{(t-1)}$ is added to next prediction through $\mathbf{W}$
- as RethinkNet polishes the prediction, difficult labels would eventually be more accurate

Example in linear case

- simple example to show how memory is used to model label correlation
  - SRN: linear transformation $\mathbf{W} \in R^{K \times K}$ and $\mathbf{U} \in R^{K \times d}$
  - no dense layer: t-th prediction $\hat{\mathbf{y}}^{(t)} = \hat{\mathbf{o}}^{(t)} = \sigma(\mathbf{U}\mathbf{x} + \mathbf{W}\hat{\mathbf{o}}^{(t-1)})$.
- $\hat{\mathbf{o}}^{(t)}[j] = \sigma(\sum_{i=1}^{d} \mathbf{U}[j, i] * \mathbf{x}[i] + \sum_{i=1}^{K} \hat{\mathbf{o}}^{(t-1)}[i] * \mathbf{W}[i, j])$
- output at t can be seems as a feature prediction + a memory prediction
- $\mathbf{W}[i, j]$ matches the correlation between i-th and j-th label



(a) memory (b) correlation

Figure: The memory transformation matrix $\mathbf{W}$ and the correlation coefficient of the yeast data set. (a) Each row of the memory weight is normalized for the diagonal element to be 1 so it can be compared with correlation coefficient. (b) Each element represents the correlation between two labels.

Cost-Sensitive Reweighting

- utilize temporary predictions to extract cost information and encode it as the importance weight of each label (intuition: if changing one label changes the cost a lot, it should be heavily weighted)
- solving a weighted binary-cross entropy easily makes RethinkNet cost-sensitive

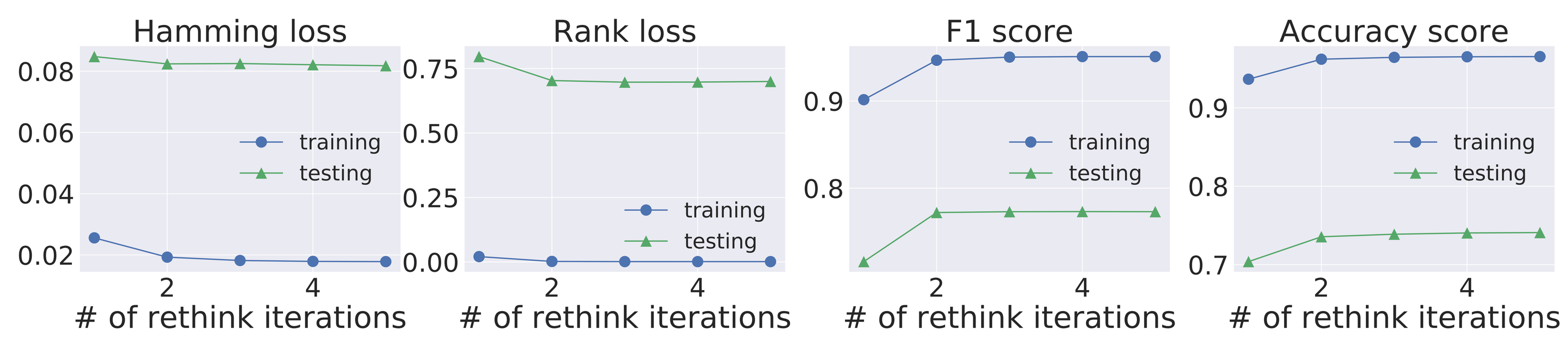
$$\frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{B}\sum_{i=1}^{K} -\mathbf{w}_n^{(t)}[i](\mathbf{y}_n[i]\log p(\hat{\mathbf{y}}_n^{(t)}[i]) + (1 - \mathbf{y}_n[i])\log(1 - p(\hat{\mathbf{y}}_n^{(t)}[i])))$$

$$\mathbf{w}_n^{(1)}[i] = 1, \quad \mathbf{w}_n^{(t)}[i] = |C(\mathbf{y}_n, \hat{\mathbf{y}}_n^{(t-1)}[i]_0) - C(\mathbf{y}_n, \hat{\mathbf{y}}_n^{(t-1)}[i]_1)|$$

Experiments

Performance w.r.t. rethink iterations

Figure: The average performance versus number of rethink iteration on the scene dataset.

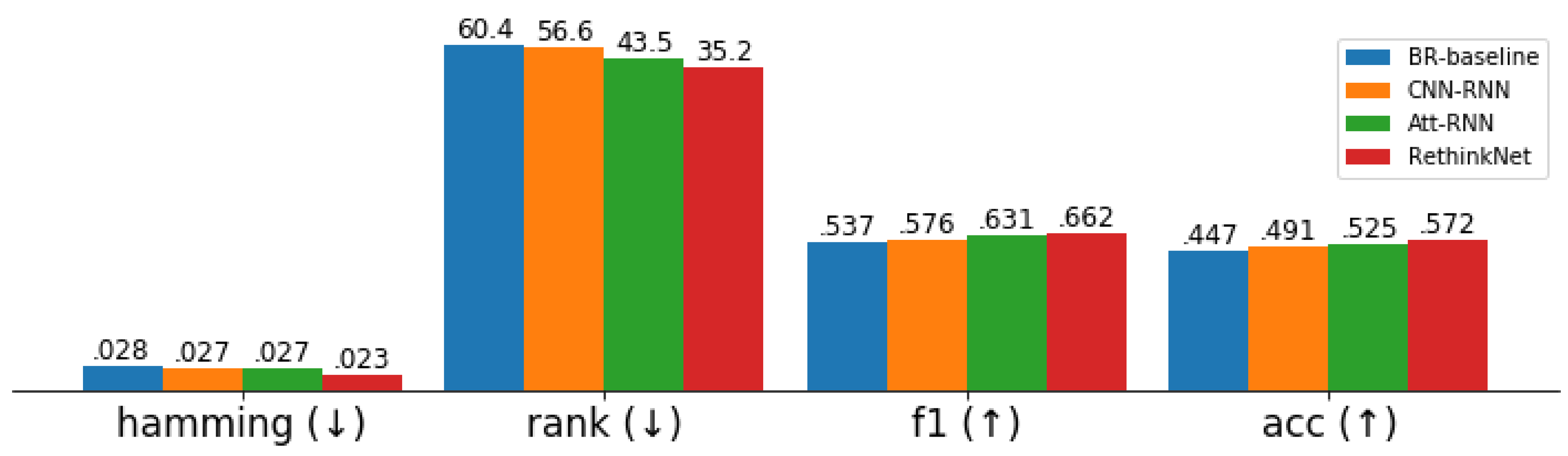


Compare RethinkNet with MLC algorithms

Table: RethinkNet versus the competitors based on t-test at 95% confidence level (win/tie/loss)

	PCC	CFT	CC-DP	CC	CC-RNN	BR
hamming (↓)	6/1/4	3/4/4	5/2/1	6/1/4	8/3/0	3/6/2
rank (↓)	5/1/5	5/2/4	7/1/0	10/1/0	10/1/0	10/1/0
f1 (↑)	6/2/3	5/4/2	5/2/1	8/3/0	10/1/0	9/2/0
acc (↑)	7/1/3	5/4/2	5/1/2	7/4/0	9/2/0	9/2/0
total	**24**/5/15	**18**/14/12	**22**/6/4	**31**/9/4	**37**/7/0	**31**/11/2

Figure: The performance of MLC algorithms on MSCOCO dataset with ResNet for feature extraction.



Conclusion

- Developed a novel MLC algorithm - RethinkNet that works well empirically.
- RethinkNet models label correlation by treating MLC problems as a sequence; this allows labels to share same amount of information, thus no label ordering problem.
- RethinkNet can be extended to be cost-sensitive with our cost-sensitive reweighting method.