Vincent Art Director – SoftRank on Image Evaluation

CHEN, Charles Liang-yu

Naver Corp.

Introduction

RESEARCH GOAL

Improve the performance of test top 50 average CTR value of image data

DESCRIPTION

This Vincent art director is trying to select good images from a bunch of inputs. That is, given some pieces of image data along with their CTR values, its goal is to evaluate which one are the best images. Therefore this issue will be mainly focusing on improving the metric of top 50 CTR average. Here we use the data set from vincent-cb001 and cab001 as our modeling data and our goal is to improve the metric from the model aspect.

DATA

Training input: 1091*3*224*224
Training CTR values: 1091 * 1
Training Labels: 1091 * 1
(derived from CTR values, will be discussed in later sessions)
Test input: 493 * 3 * 224 * 224
Test CTR values: 493 * 1

PREVIOUS WORK

From the legacy experiment results conducted by Hyojung Han, we had done binary classifications based on its top half CTRs and bottom half CTRs, and the best result we got was around 3.8% in average. Also, regressions and 5-class classification methods were also under experiment, their performances were even worse than binary classification method however. Several improvements have been done during the recent months, from research issue #547 our crew Keesoek's research result has boosts the metric up tp 4.5% to 4.8%. We will set this as the benchmark. The following will be discussing the performances of various of methods that we have implemented and its theoretical background no matter how well it performs on our dataset.

Methods and results

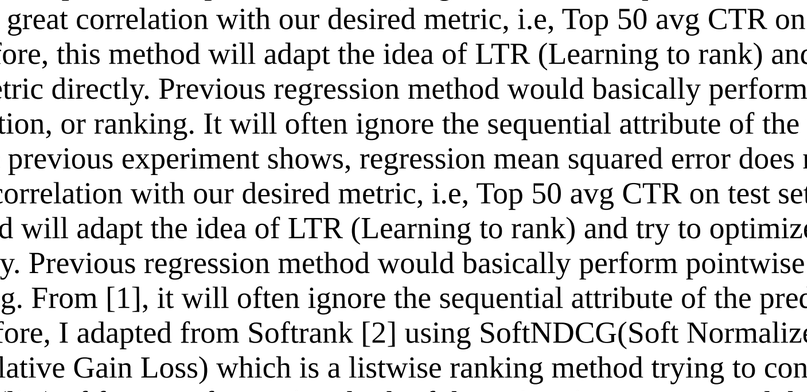
OVERVIEW

As the previous experiment shows, regression mean squared error does not seem to have a great correlation with our desired metric, i.e, Top 50 avg CTR on test set. Therefore, this method will adapt the idea of LTR (Learning to rank) and try to optimize the metric directly. Previous regression method would basically perform pointwise prediction, or ranking. It will often ignore the sequential attribute of the predicted outputs. As the previous experiment shows, regression mean squared error does not seem to have a great correlation with our desired metric, i.e, Top 50 avg CTR on test set. Therefore, this method will adapt the idea of LTR (Learning to rank) and try to optimize the metric directly. Previous regression method would basically perform pointwise prediction, or ranking. From [1], it will often ignore the sequential attribute of the predicted output. Therefore, I adapted from Softrank [2] using SoftNDCG(Soft Normalized Discounted Cumulative Gain Loss) which is a listwise ranking method trying to consider, given a batch (list) of feature of a session, both of the pointwise property and the predicted orders. Softrank in comparison is a listwise ranking method trying to consider, given a batch (list) of feature of a session, both of the pointwise property and the predicted order.

In general LTR problems, we use NDCG [3] as a metric to evaluate the information gain based on a specific query and the correlated documents, and documents with high relevance will be place in front of low relevant data, which is similar to our goal - given a set of images, we wish to find the top ones with better CTRs. However, NDCG is not a differential function and cannot be used as a loss function to our resnet. Therefore, this SoftNDCG will be the loss function that uses probabilistic model to approximate NDCG metric and can be directly optimized by a deep neural network. As a consequence, we adapted this idea into our image, and try to directly reach the goal.

FIRST TRY- DIRECTLY ADAPT SOFTNDCG AS THE LOSS

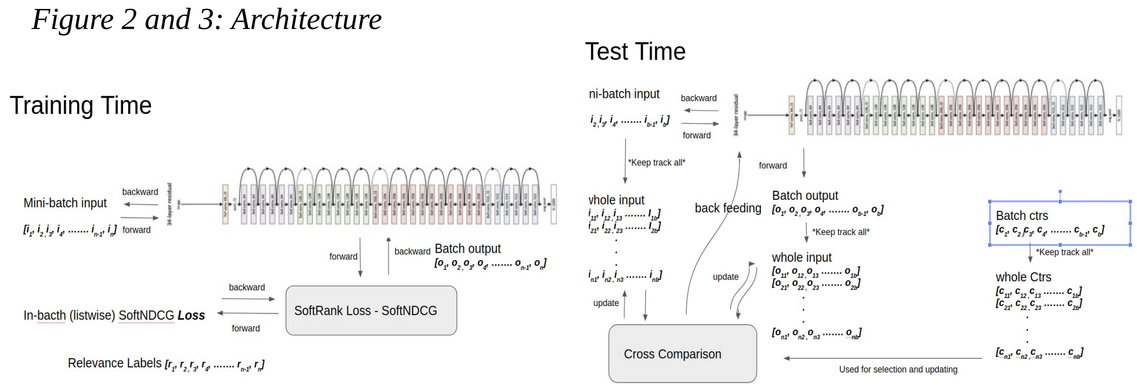
Figure 1: Architecture (For both training and testing phase)



Here we manually label data relevance with respect to their real CTR values. Based on this labels, we make our model to predict the relative order in every single list. The performance is shown as below figure 4. The best performing one reaches 4.4% of top 50 CTR average on test data, which is still slightly worse than the previous method. However, there are some architectural flaws in this version of model. Since that during either training or test set, we only preform SoftNDCG loss base on a given batch of data of size batch_size, i.e., a list, therefore our model can only learn the to rank every input of that specific size of data. It does not have a global view of the inputs. Therefore, the following paragraphs will be dealing with this problem, and boost the top 50 CTR average.

THIRD TRY – CROSS COMPARISON OF LISTS

Figure 2 and 3: Architecture

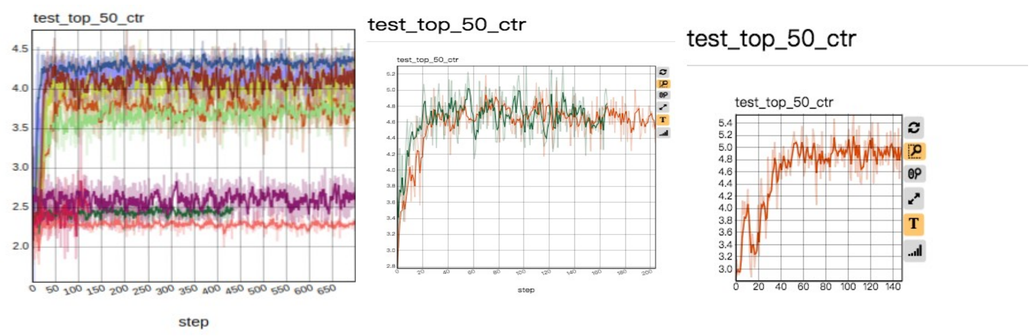


To deal with the problem mentioned above, I came up with an idea to have a better comparison towards the global distribution of predicted output. Here during the training set, we are still training model to predict a given fixed size of list of data. As this will ensure that our model has the ability to rank given any feature vectors. However, during the test time, we will select two lists of predicted outcomes, i.e., their in-batch scores of ranking, and based on the rules as below, discard half of it, and so on recursively. The structure itself is that we compare two lists of outputs, say *o1* and *o2*. Based on their predicted scores, we calculate the CTRs, *c1* and *c2* based on the order. Then, we try to find the best combination of selecting half of these two outputs. Here is the cross comparison algorithm I did:

i. If the CTR average of the combination of first halves of two lists is larger than any of its average, output the combination.
ii. If the previous rule does not apply, choose the first 3/4 of one list combine with the first 1/4 of another list, and apply the comparison of the average CTR.
ii. Otherwise output the batch with higher average CTR.

After this, we will get a list with highest combined CTRs *c3* and a list of output, say *o3*'. However, the result of output scores here may be the combination of two different previous scores, which means this new output scores is not usable anymore as it mixed up two unrelated scores. To solve this problem, I keep track of the inputs as well, so when a new higher CTR list *c3* comes out, I feed the correlated new combination of inputs (*i3*, from corresponding *i1* and *i2*) back into our trained model again to get a accurate output of the inputs, as from the paper above, our model will have the ability to rank and order any input data pieces of batch size batch_size. We do this iteration over and over again, under our buffer size is approximated to 50 as that is our goal. Each time of comparison, we pop out the data of two batch, and add the result one back. So the first level of iteration will be dealing with the first time passed data, which are not combined yet, and that will be *O(n)* complexity, which is same to regression. For the second and on iterations, as all batches are either selected or combined, therefore it requires *O(N/2), O(N/4)*... and so on, so total it is *O(N logN)* complexity for test and infer. Compare to Regression method, it is slightly slower when deploying our model. Therefore, the trade-off between the performance and time will be evaluated in later experiments. This method is still under training, result will be updated tomorrow, currently we can get an average of 4.8% ~ 5.0% on the batch size of 32. This method is still under optimization, any desired outcome will be updated soon.

Figure4 and 5 and 6: Results for First try(size 32) and Second try (size 16 and 32)



Reference:
[1] Z. Cao, T. Qin, T. Liu, and M. Tsai. Learning to Rank: From Pairwise Approach to Listwise Approach. MSR-TR-2007-40.

[2] M. Taylor, J. Guiver, S. Robertson and T. Minka, *SoftRank: Optimising Non-Smooth Rank Metrics.* Microsoft Research Cambridge, 2016.

*[3]*Y. Wang, L. Wang, Y. Li, D. He, W. Chen and T. Liu. *A Theoretical Analysis of NDCG Ranking Measures.* JMLR: Workshop and Conference Proceedings vol (2013) 1–30.