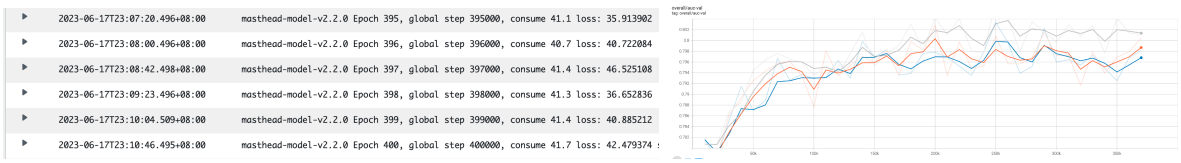
Research on Stabilized Masthead Model Training

Xuanrui Zhang

Background

During offline experiment on masthead model, we've surprisingly found that several time trials of training with same settings unexpectedly have significant AUC metrics diff on next-day's data.

This is not a normal phenomenon in the field of machine learning, as models are expected to converge after iterations of training. In other words, their training loss and evaluation metrics should be stable and models trained by multiple trials should have similar performance.



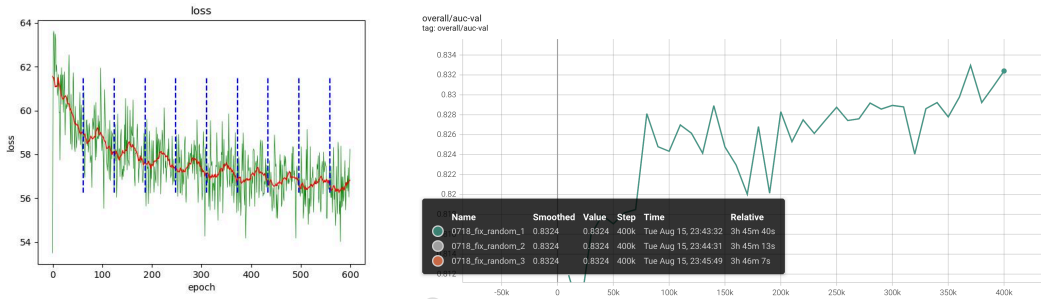
Left: training loss fluctuation; Right: unstable testing metrics

However in our case, loss is fluctuating and its performance is unstable. We recognize that this problem need to be resolved ASAP as unstably trained model could not provide reliable offline evaluation metrics and it would block further model optimization or adding new features.

Investigation

Firstly, we suspect that instability comes from some processes involving randomness, e.g. random seed in initialization, race condition by data generation with multiple processes. So we identified all places where randomness comes from and made them deterministrics by random seed setting & execution order fixing in race condition.

After making model deterministic, its evaluation metrics got stable given all inputs the same. The reproducibility could help us believe that subsequent experiment conclusions originate from new changes. However, it doesn't fully resolve our problem as training loss still fluctuates and what we want is that evaluation is stable with slight changes on input signals.



Left: training loss (red curve: smoothed loss); Right: exactly the same performance after randomness fixed

After we ruled out suspicious model randomness, we assume that there are only two possibilities remaining which might result in model instability: any part of the model which is not converged or some problematic pattern hidden in training data.

Secondly, we conducted several experiments on model structure. The first idea that comes to our mind is to simplify model structure. We implemented two simplified models: Wide&Deep and Wide-Only model, given loss value still fluctuates at the same range, we came to the conclusion that stability has nothing to do with model complexity.

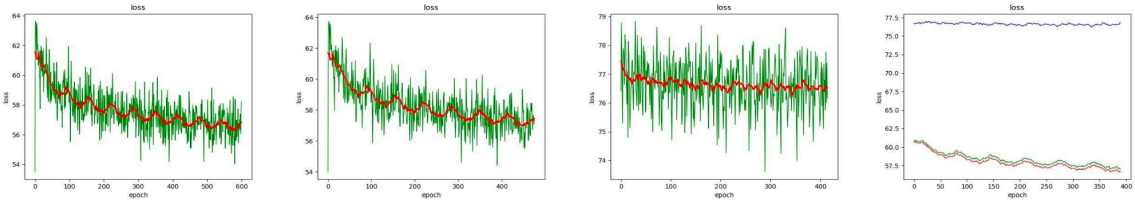


Image-1: baseline (xDeepFM); Image-2: Wide&Deep; Image-3: Wide-Only; Image-4: Comparison of above

To figure out model convergence status, we've implemented a debiased visualization algorithm which could help us observe and analyze how gradients & weights are changing during training process. We found that basically all layers are converged except layers which directly connect to some extremely sparse features, we confirmed these features were already deprecated due to business logic adjustment, thus we removed those features.

Besides, we've conducted experiments on learning rate, optimizers, gradient clipping and other hyper-parameters, results further confirmed that stability has nothing to do with model itself.

Afterwards, we focused our attention on suspicious training data as we surprisingly found that loss value has periodic oscillation on smoothed epoch-level loss, and its period length is exactly the same as steps for one iteration over training data.

Further experiment shows that model trained by date-sequentially data would result in severe loss fluctuation, thus we believe it is the sequence of training data that caused model instability. Given above insight, we fully reshuffled all data at row-level, made each iteration over training data with unique order then retrained a model. The result is promising, not only epoch-level loss gets rid of period and keeps declining but also batch-level fluctuation has reduced substantially.

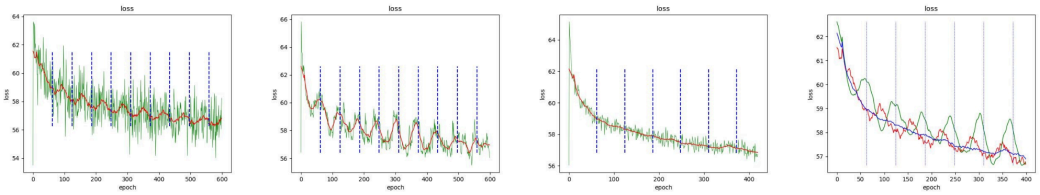


Image-1: baseline; Image-2: sequential training; Image-3: fully randomize; Image-4: Comparison of above

Based on above phenomenon, we came to the conclusion that it is the sudden change of data distribution which makes loss curvature steep that results in model instability.

Given that our training data is not sufficient as it will be iterated more than 6 times during training, which might result in overfitting problem to some extent. We dumped more 100% of user data from masthead service. Experiment confirmed that data-enriched model could increase user engagement and its embedding lookup & DNN layer converged much better.

Following is the visualization of these layers. The first row shows its weights, the earlier getting smoother the better; The second row shows its gradients, the earlier getting smaller the better.

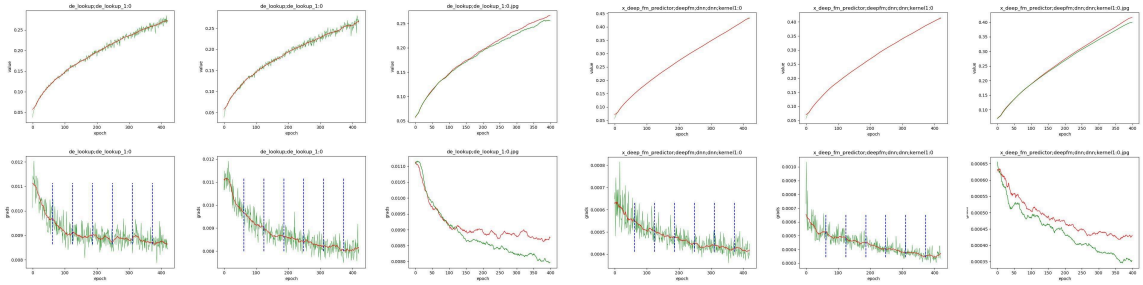
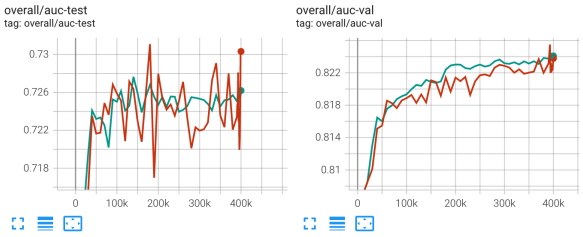


Image-1: visualization of embedding look up layer; Image-2: visualization of DNN kernel layer

Finally, we experimented with bigger batch size as batch-normalization experiment suggests that current batch size is not large enough, which would prolong model convergence. Besides, learning step is not small enough during last several epochs of model training which result in evaluation metrics keep fluctuating. It shows that doubled batch size and decay learning rate to 1% with a continuous exponential scheduler would help model converge better and improve its stability, evaluation metrics diff among multiple trials reduced to insignificant level.



stabilized eval metrics with lr decay scheduler (green curve)

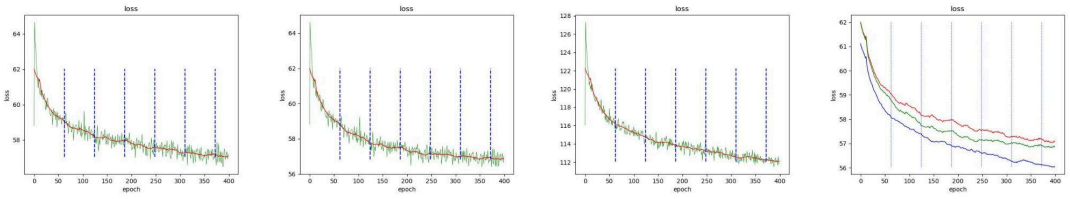


Image-1: baseline; Image-2: lr decay scheduler; Image-3: 2x batch size; Image-4: Comparison of above

Last but not the least, though instability has been resolved by full data shuffling for now, in the future we need to investigate further on how to resolve this problem by model training optimization in the case of date-sequential training data provided, as this is vital for online incremental training.

Impact

1. Realization of stabilized masthead model training
2. Training loss further reduced by reliable learning rate scheduler and bigger batch size
3. AB shows user engagement also increased: average watch_time of free user raised by 0.84%, paid user raised by 0.15%; watch through ratio of paid user significantly raised by 0.59% (p-value = 0.02)