CS410 Technology Review
Google's MultiTask Ranking System

Gargi Balasubramaniam
gargib2@illinois.edu
University of Illinois, Urbana Champaign

1 Introduction

This review motivates the problem of multi task ranking with the example of Google's Multitask Ranking system. We will first introduce the basic concepts of Multitask Learning and Ranking, and see how these concepts can be combined to understand Google's system. This review will conclude with a critique of the system and future suggestions.

2 Preliminaries

2.1 Multi Task Learning

Multi Task learning [2] is a kind of the general transfer learning scheme where we have a single input with different outputs tasks. This can be thought of as a problem where a given learner has access to multiple tasks - and the question arises as to - a) whether or not it is beneficial to simultaneously learn to perform well on all tasks, in comparison to learning independently w.r.t each task. from [1] depicts this concept.

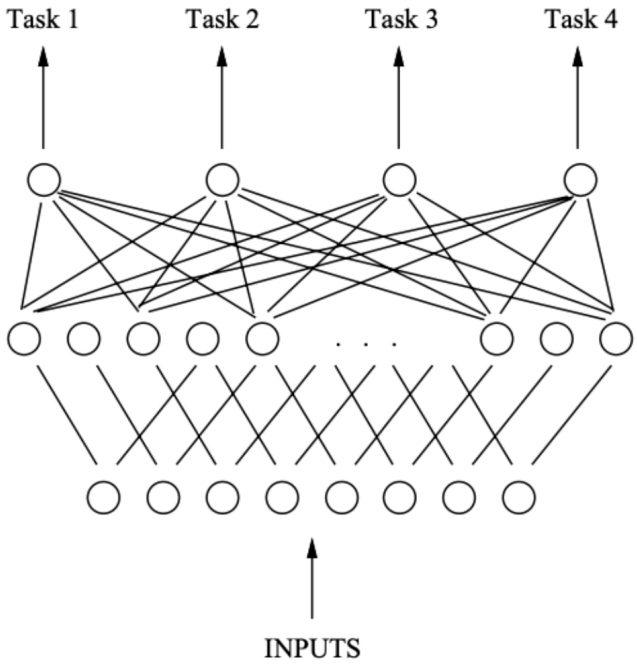

Figure 1. Multi Task Learning

2.2 Learning to Rank

The system uses a learning to rank methodology. This problem can be framed as follows: Given a training dataset with list of partially ranked items, the task is to learn to rank unseen lists in a similar fashion. The training set is typically evaluated by humans (explicit feedback) or through pooling where only a certain set of documents is ranked, and the rest is judged through implicit feedback. This can be done using proxies like number of user clicks. Learning to rank is a relevant problem in IR as number of users and their behaviour constantly keeps evolving with time. Being such a dynamic system, it is equally important for the model to be dynamic and adapt to these changes over time i.e. "learn" to rank "over time".

Further, ranking is a crucial element of the recommendation system pipeline, wherein the most relevant items are obtained by a score assigned by the ranking system. This scoring is done on a set of candidate points which are typically obtained through criteria specific to the application. For instance, Google's system uses the similarity between candidate and query video, and time-correlation of watching this video after the query video - to name a few criterion.

3 Non-triviality of problem

It is important to understand why these tasks are non-trivial, especially in the domain of video-based ecosystems like YouTube. Some challenges include:

1. One type of ranking may be motivated by different kinds of objectives, and need not always follow a single pattern (this would be inflexible, especially given the information coverage for YouTube). For example, one may like to see videos based on their history as well as based on the choice of the people who', they have subscribed to. Optimizing for multiple objectives while ranking becomes a non-trivial task in this case.
2. Supervision is required in order to evaluate the results of a given ranking system - however it is impractical to gain this feedback on a large scale. Thus, there is heavy reliance on implicit feedback mechanisms like user-clicks. Given the volatility of behaviour, these clicks need not necessarily indicate successful personalization and can introduce bias into the system.
3. YouTube contains a large corpus of information, that too in the form of videos. These videos can additionally have metadata associated like comments, likes, user similarity, etc. Thus, the scale of this problem in itself makes it non trivial to solve.

4 Google's System Overview

Any learning problem can be understood as the amalgamation of the following processes:

1. Analysis of input i.e. training Data a
2. Defining Objectives
3. Learning the model (in this case for Ranking)
4. Evaluating and correcting the model using feedback

We will now look at each of these elements one by one and understand their usage in Google's system [3].

4.1 Input Data

The input data provides two kinds of signals: proxies of user engagement such as clicks and watches, and proxies of liking the content such as likes and subscriptions. These user logs are used for training the model.

4.2 Archtecture

The authors propose using a Multi-gate Mixture of Experts (MMoE) model for modelling the multi-task objective ranking problem. Typically, Multi Task Learning systems used a shared embedding and then work through the multiple objectives independently. However, this may be too restrictive and may not allow sharing of useful information between the tasks. Thus, the authors propose using soft-parameter sharing instead of keeping them disentangled. This allows for conflicts to be modelled.

In order to mitigate the biases discussed earlier, the authors propose using a Wide and Deep model architecture. In such an architecture, typically the wider neural network is trained with the features which lead to bias. This is very similar to side tuning - a concept recently introduced in [**?**]. This allows the specific bias to be learnt and used in the final layer of the main (deep) model (thus the term wide and deep).

4.3 Evaluation

The authors evaluate their model with the AUC evaluation metric for classification and the squared loss for regression.

Drawbacks relating to multimodal feature spaces, scalability with the increase in number of objectives, and sparsity of training data are addressed.

5 Conclusion

Google's multi task ranking system is novel in the sense that they attempt to model multiple ranking objectives at the scale of YouTube, wherein the existence of multiple objectives makes natural sense. Further, they propose a method to model the selection bias in the implicit feedback and show that the their metrics increase through a live experiment.

5.1 Critique

It would be nice to see more vigorous experimentation and comparison with existing techniques

(apart from the shared bottom model) based on performance, as one cannot place this method amongst others. Further, it is unclear as to how exactly the soft parameter sharing tangibly contributes to these metrics qualitatively.

References

[1] Rich Caruana. 1997. Multitask learning. *Machine learning* 28, 1 (1997), 41–75.

[2] Sebastian Thrun. 1995. Is Learning the N-Th Thing Any Easier than Learning the First?. In *Proceedings of the 8th International Conference on Neural Information Processing Systems* (Denver, Colorado) *(NIPS'95)*. MIT Press, Cambridge, MA, USA, 640–646.

[3] Jeffrey Zhang, Alexander Sax, Amir Zamir, Leonidas Guibas, and Jitendra Malik. 2020. *Side-Tuning: A Baseline for Network Adaptation via Additive Side Networks*. 698–714. https://doi.org/10.1007/978-3-030-58580-8_41