Clipper: A Low-Latency Online Prediction Serving System

Machine learning is being deployed in a growing number of applications which demand real-time, accurate, and robust predictions under heavy query load. Within this paper, a general-purpose low-latency prediction serving system, Clipper, is proposed. This system servers as the intermediate for end-users and various machine learning frameworks. With supported caching, batching and adaptive model selection, this system fulfills the requirement of online machine learning framework serving with low latency and robust predictions.

1. The modular architecture for simplified model deployment is introduced.
2. Serval enhancement, such as caching, batching, and adaptive model selection techniques, are included, which reduce latency and improve prediction for both throughput and accuracy.
3. The system evaluation is performed on four common machine learning benchmark and a comparison between clipper and TensorFlow serving are completed.

Model abstraction and common prediction interface to simplify model deployment (for the changing state-of-art) – it is a great effort for heterogeneous model deployment, but not necessary. 1. The popularity of this system. -> maybe one main platform is enough for problem addressment. And there are possible some specific optimization for uniform eco-system, such as TensorFlow model and TensorFlow Serving. **(extend, github)** 2. The improvement in prevailing machine learning system, such as TensorFlow serving. **(extend)** 3. Different possible design target for different machine learning platform. (TensorFlow for neural network and HKT for speech recognition) There is no need for developers to develop system in different platform and perform integration.

Maximize the **use of batch (dynamic batching)** and **straggler mitigation techniques** to reduce bound tail latency - The improvement in prevailing machine learning system, such as TensorFlow serving. **(extend) Although it is interesting to the dynamic control of batch size, this technique is not unique but a borrowed notion from control theory.**

Enable online model selection for addressing different features and prediction combine - The combining of multiple models might introduce internal computation burden, comparing with one model. There is extra effort to perform model selection by introducing weights. In order to train the weighted scale (Exp3 and Exp4), multiple models require to run simultaneously, which is possible a heavy burden.

Possible scaling concern: since all the evaluation are performed in a single server and present large-scale machine learning systems are in distributed manner, there is possible scaling and communication concerns for this system.

From the experiments, the performance of TensorFlow Serving are better than this proposed system.

The focus of this system is unique for the time it published. Focusing on model deployment and serving, there are several extensions in terms of batching, caching and personalized recommendation, which compensates the shortage of other serving system at that time. The challenge-solution writing style is easy for reader to identify their contributions.

Model deployment and prediction-serving have received relatively little attention. Clipper is a layered architecture system that reduces the complexity of implementing a prediction serving stack and archives three crucial properties of a prediction serving system: low latencies, high throughputs, and improved accuracy. – two layers: model abstraction layer and model selection layer.

1. Model abstraction layer exposes a common API that abstracts away the heterogeneity of existing.
2. Model selection layer is used for dispatching queries to one or more models and combine their prediction based on feedback to improve accuracy.

Several optimizations are completed. Clipper caches predictions on a per-model basis and implements adaptive batching to maximize throughput given a query latency target. Also, for accuracy, Clipper exploits bandit and ensemble methods to robustly select and combine predictions from multiple models and estimate prediction uncertainty.

Clipper’s modular design and broad functionality impose minimal performance cost, achieving comparable prediction throughput and latency to TensorFlow Serving while supporting substantially more functionality.

Analyze the application workloads:

1. Object recognition (Three datasets for evaluation)
2. Speech recognition (large model and are often composed of many complex sub-models trained using specialized speech recognition framework, and are also personalized to different user -> use interact to help with model choice for dialects)

Comparison with Ray, TensorFlow serving

TensorFlow Serving:

TensorFlow Serving provides integration with TensorFlow models, but can be easily extended to serve other types of models. They also provide concepts like “servable” (central abstraction – a TensorFlow SavedModelBundle “Session” and a lookup table for embedding or vocabulary lookups). TensorFlow Serving can handle one or more versions of a servable over the lifetime of a single server instance. -> **enabling fresh algorithm configurations, weights, and other data to be loaded over time. Client may request for a specific version id.**