Scalable Training and Serving of Personalized Models

Daniel Crankshaw **Xin Wang** **Joseph E. Gonzalez** **Michael J. Franklin**

UC Berkeley AMPLab
{crankshaw, xinw, jegonzal, franklin}@cs.berkeley.edu

1 Introduction

The past decade has seen substantial growth in Learning Systems research [1, 9, 12, 14, 22] combining advances in system design with new efficient algorithms to enable the training of complex models on vast amounts of data. As a consequence we have seen widespread adoption of machine learning techniques to address important real-world problems [5, 17].

While this work has been wildly successful in shaping both the machine learning [19, 16, 11] and systems fields [14, 15, 1], it also ignores a big part of real-world machine learning. In particular, much of the work in Learning Systems has operated under the fiction: *the world hands me a static, potentially very large, dataset and I train an accurate, potentially complex, model.* This fiction departs from reality in two key regards that we begin to address in this work.

Changing Data: First, data is dynamic, growing over time and changing as people's interests evolve and sensing technology improves. As a consequence, we need to recognize and even embrace the notion that model training is a continuous process [19] that should reflect how the world changes. In particular, while change is constant the effect that change has on model parameters may vary substantially. For example, parameters for individual users may demonstrate heavily skewed change [4] depending on user activity while parameters associated with features that are averaged across users change more slowly. This discrepancy presents opportunities in how we split the training process across online and offline systems. Some of these changes can be modeled directly (e.g., through temporal or historical features) [6] while others are better captured as continuous learning processes. If we continue to focus our attention on model training systems that are relegated to offline or data-warehouse analytics we will struggle to address the challenges of a changing world.

Model Serving: Second, a trained model is useless unless it can be evaluated quickly and reliably. The design of learning systems does not end at ROC and speedup curves but should address the efficient and scalable evaluation of models on new inputs within the confines of stringent real-world latency requirements and while under intense query load. Intelligent services are being deployed in critical paths (e.g., making decisions on what ads to show or whether transactions are fraudulent) and they must therefore be highly available. A sudden, substantial loss in prediction accuracy or increase in latency could have substantial consequences in terms of lost revenue, risk exposure, and personal safety[18]. Model serving raises new questions around what should be materialized in advance to bring down latency and what must be computed with every query? In search of low-latency predictions what trade-offs are we willing to make in model accuracy and how can we achieve the best accuracy for a given latency on a system with highly variable load?

In this work we explore a particular formulation of multi-task learning [3] (MTL) as a mechanism to simultaneously address both the challenge of changing data and enable low-latency highly available model serving. In particular, we recognize that many applications of machine learning can no longer afford to treat users as homogeneous, learning a single model for an entire population. Instead, the shift to personalized modeling has allowed several applications to achieve state of the art accuracy[5]. We exploit this common machine learning setting to structure our multi-task learning model as treating each user as a separate task. The model then factors into two components, a shared component that averages training data across all users and thus changes slowly, and a per-task component that reflects user-specific parameters. While the initial learning formulation learns these two components jointly, we can decouple their training to perform incremental updates to only the quickly changing task-specific models.

Further, we leverage this model decomposition to design a simple serving architecture that uses the shared models to reduce prediction serving latency even for expensive models, and allows for highly dynamic models to adapt to changes in the serving system. Key to the design of the system is the observation that in order to adapt fast enough, at least some of model training must be done directly in the serving system. We train the cheap, dynamic model component - the per-user models - directly in the serving system while maintaining interactive prediction latencies and high prediction and training throughput. We exploit the fact that expensive computation is encapsulated in shared models to improve prediction latency through extensive caching, and can continue to learn on cached data without invalidating the cache. Finally we demonstrate the benefits of cheap, fast retraining directly in the serving system in an important regime: serving the first few predictions to a previously unknown user, known as the "cold-start" problem [13].

2 Problem Setup

In this work we adopt a multi-task learning (MTL) formulation to enable simultaneous offline and online learning at different rates. In the following we present our formulation of the learning objective and discuss some of the key decisions and their implications on the system design.

Given the data set $\mathcal{D} = \{(x_i, y_i, u_i)\}_{i=1}^n$ where $x_i \in \mathbb{R}^d$, $y_i \in \mathbb{R}$ (or $y_i \in \{0, 1\}$ for classification) and user $u_i \in \{1, \ldots, U\}$ our goal is to estimate a model $h(\cdot; \gamma_u)$ per user u that minimizes the regularized loss:

$$\sum_{(x,y,u)\in\mathcal{D}} L\left(y, h(x; \gamma_u)\right) + \lambda \sum_{u=1}^{U} R(\gamma_u). \tag{1}$$

where γ_u is the parameters of the model for user u, $L\left(y, h(x; \gamma_u)\right)$ is the error in the prediction, and $R(\gamma_u)$ is the regularization penalty scaled by the parameter λ.

Consistent with the standard motivation for multi-task learning, we assume shared structure across each of the learning problems (e.g., people's music preferences share a common basis in genre…) and we would like to leverage this structure to improve learning accuracy. Consequently, we decompose the parametrization of the model $\gamma_u = (w_u, \theta)$ into shared θ and user specific w_u components:

$$J = \sum_{u=1}^{U} \left(\sum_{(x,y)\in\mathcal{D}_u} L\left(y, h(x; w_u, \theta)\right) + \lambda_1 \sum_{u=1}^{U} R(w_u) \right) + \lambda_2 R(\theta), \tag{2}$$

where $\mathcal{D}_u$ is the data restricted to a particular user. This formulation is consistent with standard multi-task learning objectives, however it also exposes two new opportunities to address *continuous online learning* and *low-latency highly available model serving*. If we hold θ fixed we can decompose the problem into smaller independent learning problems enabling the system to learn each problem quickly and efficiently in an online distributed environment. In addition, we are able to learn the w_u at different rates depending on the activity of each user.

By selecting a particular structure for $h(\cdot;\ w_u, \theta)$ we can further leverage the MTL formulation to achieve improved performance and even enable a black-box architecture capable of achieving model personalization for generic model classes. Here we present two simple designs for $h(\cdot;\ w_u, \theta)$:

$$h(x;\ w_u, \theta) = \sigma\left(f(x;\ \theta) + w_u^T x\right) \qquad \text{(Multilevel Regression)}$$

$$h(x;\ w_u, \theta) = \sigma\left(\sum_{i=0}^{k} w_{u,i}^T f(x;\ \theta_i)\right). \qquad \text{(Shared Basis)}$$

where $\sigma(\cdot)$ is the identity function for regression and an inverse link function (e.g., logistic function) for classification.

Both formulations of h share an important property: *they decompose the prediction task into a user specific dot product and a user independent calculation $f(x;\ \theta)$ that can be **shared across users***. Therefore, if the evaluation of f is costly it can be cached and reused across users. Furthermore, given an oracle estimator for θ we can construct a meta algorithm (discussed below) to learn a personalized (MTL) formulation, achieving automatic task specialization of generic black box learners.

The *Shared Basis* formulation has several additional advantages. First because the treatment of x is entirely mediated through the user independent model component $f(x;\ \theta_i)$ all processing of

input (e.g., image processing, across database feature lookups and joins) can be reused across users. Second, the *Shared Basis* closely reflects the design of many feature processing pipelines and deep architectures. As a consequence, we focus our attention on the *Shared Basis* formulation in this work. However, in contrast to standard formulations of basis regression, we will assume that $w_u \in \mathbb{R}^k$ where k is typically smaller than d which enables us to more accurately estimate w_u with a few samples and to efficiently learn and manage the w_u for each user. It is also worth noting that if we constrain $w_{u_0} = 1$ and $f(x;\ \theta_i) = x$ for all $i > 0$ we recover the Multilevel regression model as a special case (though we require a more costly $k = d + 1$ user specific state).

3 Hybrid Online and Offline Learning

By applying alternating minimization on $\{w_u\}_{u=1}^U$ and θ respectively the above MTL formulation enables us to isolate the estimation of user specific and shared parameters. This is critical both from a statistical and system design perspective. In particular, from a statistical perspective we expect the estimated values of w_u to be much more sensitive to each new observation for each user (especially new users) than the estimated values of θ which depend on the entire dataset and should be more concentrated under the central limit theorem. We can leverage this statistical property in the system design by moving the estimation of θ to a periodic offline batch retraining process and continuously updating our estimates of w_u for each user as data arrives. Furthermore, because the optimal values of w_u are independent given θ this estimation process can be completed in a distributed fashion without communication or coordination.

Under the alternating minimization formulation (a form of block coordinate descent [20]) we consider two optimization algorithms: *first-order* and *oracle-mtl*. The first order method assumes that we can compute the gradients:

$$\frac{\partial J}{\partial w_u} = \sum_{(x,y)\in D_u} \frac{\partial L}{\partial h}\frac{\partial h}{\partial w_u} + \lambda_1 \frac{\partial R(w_u)}{\partial w_u} \tag{3}$$

$$\frac{\partial J}{\partial \theta_i} = \sum_{u=1}^{U} \sum_{(x,y)\in D_u} \frac{\partial L}{\partial h}\frac{\partial h}{\partial \theta_i} + \lambda_2 \frac{\partial R(\theta)}{\partial \theta_i} \tag{4}$$

If we assume the *Shared Basis* formulation for h we can eliminate the need for the user to provide Eq. 3 which can be directly computed by the system under a range of loss and regularization choices. Furthermore, with a squared loss and regularization we can apply rank-one updates to efficiently maintain the optimal w_u in closed form.

However, to recompute θ offline in the *first-order* formulation we would need to limit ourselves to general purpose first-order solvers. For certain choices of f there may be optimized off-the-shelf black box solvers (e.g., deep learning). To address this we propose an *oracle-mtl* formulation which relies on an optimal black box solver (oracle),

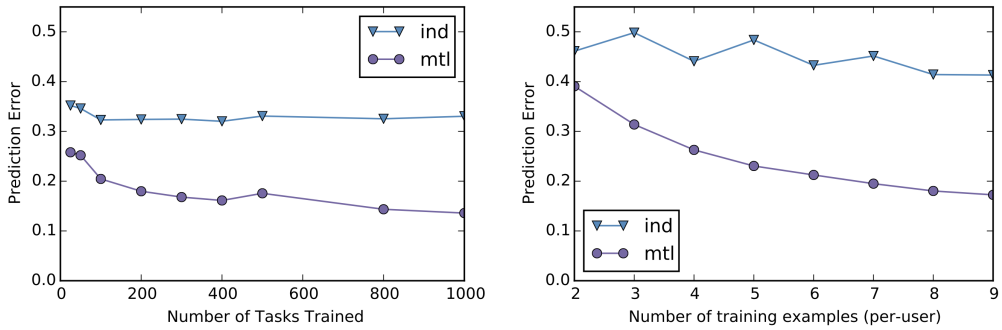
$$\arg\min_{\theta} L\left(y, f(x;\ \theta)\right) + \mu R(\theta), \tag{5}$$

to estimate the optimal θ holding w_u fixed. However to use the *oracle-mtl* formulation we currently need to impose constraints on loss (squared) to enable construction of weighted datasets that incorporate the dependence on w. We are currently exploring methods around MTL boosting [5] to generalize this construction to other loss functions.

4 Early Results

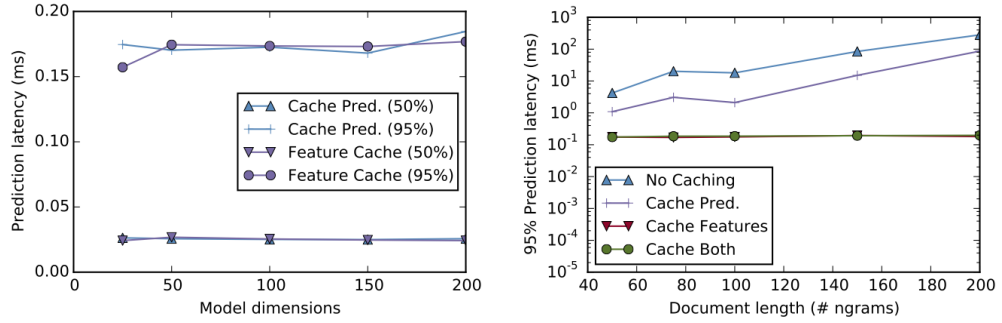
We evaluate the accuracy improvements of the MTL formulation in two settings: exploitation of shared training data for improved accuracy across tasks (Figure 1a) and the ability of the shared feature models to generalize to the cold-start setting (Figure 1b). In both settings, we see that the MTL formulation learned via the oracle meta-algorithm substantially outperforms learning independent models for each task. In Figure 1b, we see that the shared features allow the new task models to learn from limited training data much more effectively than the much higher-dimensional independent task models, substantially improving performance on cold-start.

We next evaluate the effectiveness of various forms of caching on improving prediction latency. We investigated the effect of both caching the evaluation of shared features models and caching task-specific predictions for two model implementations (see Figure 2). In both settings, a collaborative



(a) **Effect of task pool size on accuracy** (b) **Cold-Start Accuracy**

Figure 1: Benefits of multi-task learning on prediction accuracy



(a) **Collaborative Filtering** (b) **Text Recommendation**

Figure 2: Comparison of various caching strategies on prediction latency

filtering setting where feature computation is cheap and a document recommendation setting where computing the features involves computing trigrams over the document, the benefit of prediction caching in addition to feature-caching is negligible. This benefit is small because the additional cost of computing a prediction once the features have been computed is small by design. However, as Figure 2b demonstrates, in settings where feature computation is expensive the ability to effectively cache features becomes significant. The feature cache is effective for three reasons: the result of feature computation is small even when intermediate results are large, the feature cache is shared among all tasks leading to a high cache hit rate, and online updates are performed on top of the cache leading to a low cache-invalidation rate.

5 Related Work

This work builds on related work in the machine learning and database systems communities. Most closely related, is the work of Agarwal et al. [1] and our earlier work on the Velox model-serving system [7]. The work by Agarwal et al. describes the design of LASER, the general purpose machine learning framework developed to train and serve models at LinkedIn. Likewise the work on Velox describes an early prototype of a model serving framework developed as part of the Berkeley Data Analytics stack.

To support personalized modeling, both introduced a multitask formulation that leverages statistical sharing across users. The work by Agarwal et al. focuses on a restricted class of regularized bi-variate linear models, while Velox considered a restricted version of the *Shared Basis* model formulation. Inspired by these earlier systems, we introduced a general framework that is capable of expressing both the LASER and Velox model formulations as well as a broader class of bandit algorithms needed to support competing feature functions.

To address latency both the LASER and Velox systems focused primarily on caching. The work by Agarwal et al. proposed a richer feature cache capable of caching at many stages within a feature function as well as a similar anytime feature evaluation strategy to the one adopted here. In contrast

this work, introduces cache equivalence classes and feature coarsening to tradeoff a small reduction in accuracy for a substantial improvement in cache efficiency.

Each of these works differ considerably in how they approach learning. The work by Agarwal et al. focuses on batch distributed offline retraining of both task specific and shared weights using ADMM. Velox suggested a similar hybrid online and offline approach to training but does not provide an algorithm. In this work we explore hybrid online and offline learning with both supervised and unsupervised feature training. In the supervised setting we leverage additional side information (e.g., , the topic of the news article) to construct predictive features. In the unsupervised setting we describe a simple alternating minimization heuristic capable of automatic personalization of generic black-box learning algorithms.

In the context of more general database systems this work builds on the the work of MauveDB [8] and LongView [2] which first proposed model serving and management within data-management systems. MauveDB first cast the the inference task as views on a model explored various materialization and caching strategies including materialization and caching of intermediate state analogous to our feature caching. LongView explored a similar setting but introduced a query optimizer capable of trading off accuracy and computationally efficiency.

In the machine learning community perhaps the most closely related work is in the context of multi-task learning. Yu et al. [21] introduced an alternating formulation of multi-task learning for scalable inference based on a bilinear model similar to that of [1]. Stern et al. [19] introduced a scalable Bayesian framework for multitask learning that was later extended by Graepel et al. [10] to manage ad serving for Bing search. Similar to our work, this work also addressed evolving user interests. However in all three cases the system design was largely focused on the offline setting.

6 Conclusion and Ongoing Work

This research project explores the challenges around online model serving and training. In this work we begin to explore how an MTL formulation can be used to address both statistical and computational challenges. Our preliminary results demonstrate the potential gains in accuracy and reduction in latency associated with MTL based learning in the online setting. We are in the process of extending our analysis of the oracle training formulation to boosted MTL and generalizing our system to support a much richer family of models h.

Acknowledgments

This research is supported in part by NSF CISE Expeditions Award CCF-1139158, DOE Award SN10040 DE-SC0012463, and DARPA XData Award FA8750-12-2-0331, and gifts from Amazon Web Services, Google, IBM, SAP, The Thomas and Stacey Siebel Foundation, Adatao, Adobe, Apple, Inc., Blue Goji, Bosch, Cisco, Cray, Cloudera, EMC2, Ericsson, Facebook, Fujitsu, Guavus, HP, Huawei, Informatica, Intel, Microsoft, NetApp, Pivotal, Samsung, Schlumberger, Splunk, Virdata and VMware.

References

[1] D. Agarwal, B. Long, J. Traupman, D. Xin, and L. Zhang. Laser: A scalable response prediction platform for online advertising. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*, WSDM '14, pages 173–182, New York, NY, USA, 2014. ACM.

[2] M. Akdere et al. The case for predictive database systems: Opportunities and challenges. In *CIDR*, 2011.

[3] R. Caruana. Multitask learning. *Mach. Learn.*, 28(1):41–75, July 1997.

[4] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon. I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system. In *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, IMC '07, pages 1–14, New York, NY, USA, 2007. ACM.

[5] O. Chapelle, P. Shivaswamy, S. Vadrevu, K. Weinberger, Y. Zhang, and B. Tseng. Boosted multi-task learning. *Mach. Learn.*, 85(1-2):149–173, Oct. 2011.

[6] W. Chu, L. Li, L. Reyzin, and R. E. Schapire. Contextual bandits with linear payoff functions. In *International Conference on Artificial Intelligence and Statistics*, pages 208–214, 2011.

[7] D. Crankshaw, P. Bailis, J. E. Gonzalez, H. Li, Z. Zhang, M. J. Franklin, A. Ghodsi, and M. I. Jordan. The missing piece in complex analytics: Low latency, scalable model management and serving with velox. In *CIDR 2015, Seventh Biennial Conference on Innovative Data Systems*, 2015.

[8] A. Deshpande and S. Madden. MauveDB: Supporting model-based user views in database systems. In *SIGMOD*, 2006.

[9] J. E. Gonzalez et al. Powergraph: Distributed graph-parallel computation on natural graphs. In *OSDI*, 2012.

[10] T. Graepel, J. Q. Candela, T. Borchert, and R. Herbrich. Web-Scale Bayesian Click-Through rate Prediction for Sponsored Search Advertising in Microsoft's Bing Search Engine. *ICML*, pages 13–20, 2010.

[11] U. Irmak and R. Kraft. A scalable machine-learning approach for semi-structured named entity recognition. In *Proceedings of the 19th international conference on World wide web*, pages 461–470. ACM, 2010.

[12] T. Kraska, A. Talwalkar, J. C. Duchi, R. Griffith, M. J. Franklin, and M. I. Jordan. Mlbase: A distributed machine-learning system. In *CIDR*, 2013.

[13] X. N. Lam, T. Vu, T. D. Le, and A. D. Duong. Addressing cold-start problem in recommendation systems. In *Proceedings of the 2Nd International Conference on Ubiquitous Information Management and Communication*, ICUIMC '08, pages 208–211, New York, NY, USA, 2008. ACM.

[14] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su. Scaling distributed machine learning with the parameter server. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pages 583–598, Broomfield, CO, Oct. 2014. USENIX Association.

[15] L. Morrisroe, J. Connelly, J. Everett-Church, Q. Lu, S. Milano, D. Shen, and S. Wong. Method and system for serving advertisements, June 30 2005. US Patent App. 10/928,532.

[16] M. Nickel, V. Tresp, and H.-P. Kriegel. Factorizing yago: Scalable machine learning for linked data. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 271–280, New York, NY, USA, 2012. ACM.

[17] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, and M. Young. Machine learning: The high interest credit card of technical debt. In *SE4ML: Software Engineering for Machine Learning (NIPS 2014 Workshop)*, 2014.

[18] D. Sculley, M. E. Otey, M. Pohl, B. Spitznagel, J. Hainsworth, and Y. Zhou. Detecting adversarial advertisements in the wild. *KDD*, pages 274–282, 2011.

[19] D. H. Stern, R. Herbrich, and T. Graepel. Matchbox: large scale online bayesian recommendations. *WWW*, pages 111–120, 2009.

[20] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *J. Optim. Theory Appl.*, 109(3):475–494, June 2001.

[21] P. J. Yu, Hsiang-Fu, P. Kar, and I. S. Dhillon. Large-scale Multi-label Learning with Missing Labels. *ICML*, pages 593–601, 2014.

[22] M. Zaharia et al. Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *NSDI*, 2012.