Scalable Hands-Free Transfer Learning for Online Advertising

Brian Dalessandro
Daizhuo Chen
Dstillery
470 Park Ave S
New York, NY 10016
briand,hchen@dstillery.com

Troy Raeder
Claudia Perlich
Melinda Han Williams
Dstillery
troy,claudia
melinda@dstillery.com

Foster Provost
New York University & Dstillery
44 W. 4th St
New York, NY 10012
fprovost@stern.nyu.edu

ABSTRACT

Internet display advertising is a critical revenue source for publishers and online content providers, and is supported by massive amounts of user and publisher data. Targeting display ads can be improved substantially with machine learning methods, but building many models on massive data becomes prohibitively expensive computationally. This paper presents a combination of strategies, deployed by the online advertising firm Dstillery, for learning many models from extremely high-dimensional data efficiently and without human intervention. This combination includes: (i) A method for simple-yet-effective transfer learning where a model learned from data that is relatively abundant and cheap is taken as a prior for Bayesian logistic regression trained with stochastic gradient descent (SGD) from the more expensive target data. (ii) A new update rule for automatic learning rate adaptation, to support learning from sparse, high-dimensional data, as well as the integration with adaptive regularization. We present an experimental analysis across 100 different ad campaigns, showing that the transfer learning indeed improves performance across a large number of them, especially at the start of the campaigns. The combined "hands-free" method needs no fiddling with the SGD learning rate, and we show that it is just as effective as using expensive grid search to set the regularization parameter for each campaign.

Categories and Subject Descriptors

I.5.4 [**Computing Methodologies**]: [Pattern Recognition - applications]

Keywords

Transfer Learning; Stochastic Gradient Descent; Adaptive Learning Rates; Adaptive Regularization

1. INTRODUCTION

Online display advertising, served through real-time bidding systems, is a major source of revenue for web publishers. As media consumption continues to shift to web-hosted content, advertising dollars continue to pour into the system. Global display advertising revenues exceed 40 billion USD annually are growing at double-digit rates [7]. Real-time bidding systems are run by *ad exchanges*: auction platforms connecting sellers of ad placements (usually web publishers with ad space to monetize) and buyers (usually independent firms like Dstillery,[1] operating on behalf of consumer brands and their agencies). The goals of the buyers vary. At Dstillery, our ultimate goal is to identify the *best* possible audience for each of our individual advertising clients and advertise to this audience. What exactly "best" means depends upon the goals of the advertiser, but usually we optimize some form of *post-view conversion rate*. The definition of *conversion* is again campaign-specific, but always requires taking some action (such as visiting a site, buying a product, or signing up for a service). The *post-view* qualifier means the conversion is observed in some interval following an ad exposure but without the necessity of an ad click.

The data that we use for modeling is discussed in detail in Section 2, but the foundation of our targeting methodology is a suite of methods for high-dimensional, sparse classification (ranking) that builds multiple models for each advertising campaign individually. Training a single large-scale predictive model is no longer a rare feat. Our machine learning system trains, deploys, and continually retrains *thousands* of predictive models simultaneously. The performance requirements of display advertising customers and the need to operate many models simultaneously pose a number of technical challenges, which when combined motivate an efficient "hands-free" operation—robust performance with minimal intervention by the data science team. The specific challenges we consider are:

1. **Cold Start**: Targeting must perform well from the beginning of a campaign, even though there may be very little or even *ZERO* post-impression conversion data.
2. **Non-Stationarity**: Human browsing behavior, the basis of our models, changes over time due to natural seasonality and marketer-specific external factors.
3. **Consistency**: Since we restrict ourselves from pooling conversion data across advertisers (due to their com-

[1]Formerly Media6degrees

petitive concerns), we build separate models for each one. Each model must perform well in spite of large disparities in conversion rate and audience size.

4. **Scale and Robustness**: The system builds and maintains many different predictive models for each of hundreds of campaigns simultaneously. These models have a limited number of data scientists to monitor and validate them. Thus we need an automated and robust hands-free approach that can still learn the idiosyncrasies of individual campaigns.

Our current system addresses these issues with a number of different components, amply described in prior work (see [11] and the citations therein). The current paper explores an alternative, novel solution to the same goal by tightly integrating transfer learning [10] and scalable predictive modeling. Recent developments in adaptive learning-rate schedules [14] and adaptive regularization [13] allow for incremental training of linear models in millions of dimensions without exhaustive hyper-parameter searching. Our complications are (i) the need of transfer learning to support cold start and few available positive examples, and (ii) the application to ultra-high-dimensional modeling. This work extends the prior work first by incorporating the transfer learning steps directly into the large-scale linear models using Bayesian logistic regression (section 3) and second by combining and extending state-of-the-art methods to enable hands-free, scalable stochastic gradient descent (SGD, Section 4). The combination of these methods demonstrably improves campaign performance, especially in the early stages of the campaign, and this improvement is achievable within the throughput limitations of the system. We will discuss more specific related work in the relevant sections. We are not aware of a prior paper that describes and evaluates a production-ready combination of current state-of-the-art learning methods for efficient, hands-free transfer learning of large numbers of predictive models from extremely sparse data.

Specifically, the contributions of this paper are:

1. We develop a transfer learning system based on these methods and show that combining data from general conversions with data on post-impression conversions improves targeting performance across a large number of campaigns.
2. We derive a new update rule for the adaptive learning-rate algorithm of [14] so that it supports sparse updates and is mathematically compatible with the adaptive regularization algorithm of [13].
3. We provide an algorithm for using both adaptive learning rates and adaptive regularization simultaneously. We show that, within this framework, adaptive regularization is just as effective as an expensive grid search for the regularization parameter.
4. We use production data to show the effect of transfer learning "in vivo", specifically focusing on its ability to address the cold start problem for new campaigns. (Long-running campaigns have more natural training data and therefore derive less benefit from transferring information from other distributions.)

The remainder of the paper shows how these and other developments in linear modeling can be combined into a continuously updating, hyperparameter-agnostic ad-targeting system that transitions from learning on a proxy population (*all* conversions) to the true population (*post-impression* conversions) as more data becomes available. Our results show that this strategy is indeed effective for improving campaign performance, especially in the early stages of the campaign, and this improvement is more scalable than alternative methods.

2. DATA COLLECTION PROCESS

The data which informs our proprietary ad-targeting models comes primarily from multiple continuous data streams. One stream records general *browsing* events of users as they consume content across the web. Another similar stream records interactions between users and the brands we represent. These interactions include home page visits and conversions on their owned websites. The last data stream we consider is the actual ad serving process. We serve ads to users based on various criteria and following the ads we record whether or not these users convert. From these data streams we can construct different but related classification (ranking) tasks.

Our most obvious data asset for campaign optimization is the campaign data itself. If we want to draw training data directly from the target distribution, we can randomly target users with impressions, subsequently labeling a user as positive if she converts soon after exposure to the ad and negative otherwise. However, this poses serious challenges. Serving ads randomly at a necessary scale—to avoid the selection bias inherent in training on data resulting from targeted impressions—is something our stakeholders are often reluctant to do. Moreover, conversion rates are generally quite low (often 10^{-6} or less), which leads to the need to serve massive number of randomly targeted ads just to obtain a moderate number of conversions for training.

This paper focuses on a method to leverage other data sources for cheap and reliable "adjunct" training data. The Bayesian transfer learning approach provides a principled method for combining these data with a smaller amount of randomly sampled post-impression data.

Before a client initiates an advertising campaign, we place tracking pixels on various parts of the client's web site. Visits to these pixeled pages get logged (anonymously) for analysis and optimization. In parallel, we collect web browsing data on these same anonymous users from data partners across the web. This data is stored in our system as a set of anonymized (hashed) URL strings linked to a user's cookie. These two mechanisms, which are independent of actually serving ads, when combined give us both features and labels that we can use to build the adjunct training data. Specifically, we build training instances in a *backward looking* fashion. For a given period of time, we select all pixel events coming from the client's website and label the corresponding users as positive instances. We then select a random subset of users that were active in the same period but did not incur a pixel event and label them as negative instances. This design choice is very practical: while post-impression conversion events are typically rare, general visits or conversions on the client's website are orders of magnitude more abundant.

Thus we have two data sets for each campaign—one drawn from the target application's distribution, but which comprises relatively few positive instances, and another that is naturally more abundant and cheaper, but doesn't exactly represent the problem we are trying to solve. The challenge addressed in this paper is to use these data to build models to estimate the relative likelihood that a previously unseen user will convert after seeing an ad. The rest of the paper describes and evaluates a Bayesian transfer learning framework implemented to achieve this, as well as the modern machine learning methods we have combined and enhanced to do so at scale.

3. SCALABLE TRANSFER LEARNING

Our goal, in a nutshell, is to improve the performance of conversion-rate modeling at all points of the campaign. We do this by first initializing our model with a learned prior and then incrementally improve the prediction task by incorporating post-impression conversions as they become available. Prior work uses the term *transfer learning* [10] to refer to any situation where a model learned on one classification task (called the *source* task) is used to improve a predictive model applied to a different task, the *target* task. We adopt the language and notation from [10] to present our version of transfer learning more formally.

Formally, a *classification task* $\{\chi, P(X), \gamma, P(Y|X)\}$ is composed of a feature space χ, a probability distribution $P(X)$, where $X = \{x_1, x_2, ..., x_d\} \in \chi$ is a feature vector, a label space γ, where $Y \in \gamma$ is the outcome of interest and lastly, a corresponding objective function $P(Y|X)$. In our particular situation, we refer to the campaign data as the *target* and the auxiliary data described in section 2 as the *source*. We generally assume that the source and target tasks share the same feature space χ but generally assume that $P^S(X) \neq P^T(X)$ for all X.[2] We also have different outcome spaces γ for both data sets, though they are related. In our domain γ^S represents conversion labels derived from observing general traffic on the client's website whereas γ^T is derived from post-impression conversions.

Our main estimation problem is to learn $P^T(Y^T|X)$ from the target data. We do this by first learning $P^S(Y^S|X)$ from the source data and then use that estimator as a prior on our estimator for $P^T(Y^T|X)$. We generally assume that $P^T(Y^T|X) \neq P^T(Y^S|X)$, but that the structure of the estimators (specifically the weight vectors from linear models) are similar and that is why we can use the source model as a prior on the target.

To put this more formally, consider a logistic regression model for a data set $D^S = \{(X_1, y_1^S), \ldots, (X_N, y_N^S)\}$. We use the standard log-likelihood formulation and denote the sample logistic loss function as

$$\ell_t(\beta) = -y_t \log p_t(\beta) - (1 - y_t)\log(1 - p_t(\beta)), \quad (1)$$

where $p_t(\beta) = (1 + e^{-\beta X_t})^{-1}$. Again we use superscript S and T to differentiate between the source and target data. Given the source data, we define what we call the source model:

$$\hat{\mu}^S = \arg\min_{\mu} \sum_{t=1}^{N^S} [\ell_t^S(\mu) + \lambda^S r(\mu)],^{3} \quad (2)$$

where $r(\mu)$ is a suitable regularization function and λ^S a regularization weight.

[2]When referring to constructs within target or source data, we use the superscripts S and T to differentiate

[3]We use μ in this equation instead of β to distinguish it as the initialization model learned on source data

We expect that the source model is relatively effective for identifying a high-performance audience even though the training data are not drawn from the "proper" (target) distribution. We want a model that will rank highly a set of users who are likely to convert following the presentation of an ad. The source model instead identifies users who are similar to a client's customer set with respect to their web browsing history, independent of the advertising process.

We can directly model the target task by first creating an appropriate dataset. Let the target data set $D^T = \{(X_1, y_1^T) \ldots (X_N, y_N^T)\}$, where X_t is defined as above and y_t^T is defined as a post-impression conversion. In order to induce transfer learning from $P^S(y^S|X)$ to $P^T(y^T|X)$, we add a non-zero Gaussian or Laplace prior on the logistic regression coefficients that estimate $P^T(y^T|X)$. Our resulting optimization problem then looks like:

$$\hat{\beta}^T = \arg\min_{\beta} \sum_{t=1}^{N^T} [\ell_t^T(\beta) + \lambda^T r(\beta - \hat{\mu}^S)], \quad (3)$$

where again we are using the standard logistic loss. The differences between equations (2) and (3) are that in (3) we are training over the target data and have modified the regularization function in (3) to incorporate a non-zero informative prior. This form of knowledge transfer is highly scalable, because although we do incur the additional cost of learning μ^s, the cost of learning with equation 3 is no more than had we used a zero-mean prior. This is a major benefit in support of this method over more complicated transfer learning methods [10] and a motivating factor for putting it to use in our system.

Incorporating a source model for transfer learning as a Bayesian prior is not new. In a text-mining application, Chelba et al [5] successfully incorporate a prior model as a Bayesian prior into a maximum entropy Markov model to recover the correct capitalization of uniformly-cased text. Similarly, Arnold et al [2] use the hierarchical structure of features to learn informative priors over features to use in training conditional random fields. For transfer learning with logistic regression, Raina et al [12] perform text classification using a learned Gaussian prior. However, the formulation is quite different; we view our formulation as much more straightforward (they focus on the prior variance). In advertising applications, Chapelle et al [4] use informative priors in logistic regression models but not explicitly as a transfer learning mechanism. In their application they build models incrementally and use the previous iteration's model as the prior. We are not aware of any prior work that explicitly uses Bayesian transfer learning in a large-scale production application.

4. HYPERPARAMETER FREE LEARNING

Recall that our goal is twofold. As described above, we would like to take advantage of both the plentiful source data and the expensive-but-more-appropriate target data. We also would like to do this efficiently with minimal human curation. A popular and effective algorithm for building linear models with large-yet-sparse, data is Stochastic Gradient Descent (SGD) [4, 9, 15]. The main drawback of SGD is that it includes various hyper-parameters (learning rates, regularization parameters, etc.) that need to be carefully tuned in order to achieve good performance. Parameter tuning ultimately complicates and lengthens the model training phase. Firms that need to build hundreds or thousands of models simultaneously, in production, face a daunting task setting these hyper-parameters separately for each campaign.

To enable building many models with few data scientists, and to reduce the training load on our system, we have developed a "hands-free" approach that does not require either manual parameter setting or expensive grid-search. The methods we present here are well suited for large and sparse consumer behavior data and they integrate well with the transfer-learning described above. In particular, we present a hyper-parameter-free SGD logistic regression training algorithm based on combining and enhancing two recent developments: the adaptive learning rates [14] (which we call *NoPesky*) and adaptive regularization [13].

We made several enhancements to make NoPesky learning rates and adaptive regularization work together and efficiently in the current context. Specifically, we adapted the NoPesky learning rate method so that we can 1) take advantage of the sparse structure of the data to run efficient SGD updates, 2) explicitly integrate the regularization term in order to make NoPesky work with adaptive regularization, and 3) give the NoPesky algorithm a more robust starting point when positive samples are rare.

It is helpful to start by considering the update step of a standard SGD algorithm: at time t, we observe a new training sample with per-sample loss

$$f_t(\beta) = \ell_t(\beta) + \lambda r(\beta - \mu), \tag{4}$$

where β comprises the model coefficients; ℓ_t is the per-sample logistic loss function as defined in Section 3; λ is the regularization factor, and r is the regularization function having the form $r(\beta - \mu) = \frac{1}{2}\sum_{i=1}^{d}(\beta_i - \mu_i)^2$. We denote the average loss function by $f = \frac{1}{N}\sum_{t=1}^{N} f_t$, and the average logistic loss function by $\ell = \frac{1}{N}\sum_{t=1}^{N} \ell_t$. With each training sample, and for each dimension i of the model, we calculate the gradient $\nabla_{t,i} = \frac{\partial f_t}{\partial \beta_i}\mid_{\beta=\beta_t}$ and update the dimension's coefficient via

$$\beta_{t+1,i} = \beta_{t,i} - \eta_{t,i}\nabla_{t,i}, \tag{5}$$

where $\eta_{t,i}$ is the learning rate at time t for dimension i.[4] Because our algorithm works with both source and target models, we drop the superscripts S and T throughout this section.

4.1 NoPesky Learning Rates

The NoPesky learning rate method sets the SGD learning rates in a greedy manner by choosing a rate that minimizes the expected value of the average loss function f after each SGD update. It is generally impossible to solve this minimization problem exactly, but an approximate solution can be obtained and works well in practice. The method utilizes the second-order derivatives of the per-sample loss function, making it suitable for learning models with a smooth loss function, such as linear and logistic regression. Our experiments validate prior research [14], showing that SGD algorithms trained with the NoPesky learning rates perform as well as state-of-the-art adaptive learning rate schedules such as AdaGrad [6]. The difference is that NoPesky learning require no systematic parameter tuning and thus can promote faster learning.

We first sketch the core steps of the original NoPesky learning rate method that are necessary for presenting our adaptations. The method treats the stochastic gradient $\nabla_{t,i}$ as a random variable, and chooses a learning rate to minimize the expected regularized loss after one SGD update, that is:

$$\eta_{t,i} \leftarrow \arg\min_{\eta} E_{\nabla_{t,i}}[f(\beta_{t,i} - \eta\nabla_{t,i}e_i)], \tag{6}$$

where e_i is a unit vector in dimension i. In order to solve the above minimization problem, three approximations are made. First, the average loss function f is approximated by a quadratic function in the neighborhood of β_t. This is achieved by approximating its first-order derivative by $g_{t,i} \approx E[\nabla_{t,i}]$, and the second-order derivative by $h_{t,i} \approx E[\frac{\partial^2 f_t}{\partial \beta_i^2}\mid_{\beta=\beta_t}]$. Second, the first-order moment of $\nabla_{t,i}$ is approximated by $g_{t,i} \approx E[\nabla_{t,i}]$. And lastly, the second-order moment of $\nabla_{t,i}$ is approximated by $v_{t,i} \approx E[\nabla_{t,i}^2]$. One can then solve (6) and obtain an approximate solution, which is set as the per-sample, dimension specific learning rate.

$$\eta_{t,i} = \frac{1}{h_{t,i}}\frac{g_{t,i}^2}{v_{t,i}}. \tag{7}$$

The three approximate variables are calculated as running averages

$$\begin{aligned} g_{t,i} &= (1-\tau_{t,i}^{-1})g_{t-1,i} + \tau_{t,i}^{-1}\nabla_{t,i}, \\ v_{t,i} &= (1-\tau_{t,i}^{-1})v_{t-1,i} + \tau_{t,i}^{-1}\nabla_{t,i}^2, \\ h_{t,i} &= (1-\tau_{t,i}^{-1})h_{t-1,i} + \tau_{t,i}^{-1}\left.\frac{\partial^2 f_t}{\partial \beta_i^2}\right|_{\beta=\beta_t}, \end{aligned} \tag{8}$$

where $\tau_{t,i}$ is an averaging window, that can be heuristically set by $\tau_{1,i} = 1$ and

$$\tau_{t+1,i} = (1 - \frac{g_{t,i}^2}{v_{t,i}})\tau_{t,i} + 1. \tag{9}$$

The use of a dynamic averaging window determined by equation (9) to update equation (8) enables this method to adapt to non-stationary data. When the data distributions change, τ is automatically reset to a lower number which increases the learning rate and enables more exploration.

To work within our system, we have made the following enhancements to the NoPesky method.

Sparse Updates.

As discussed above, online webpage visitation data is extremely sparse. It is desirable to take advantage of this extreme sparsity to speed up the SGD updates—specifically, we would like to enhance the NoPesky method such that at step t, we only update dimension i if $x_{t,i} \neq 0$.

In regularized logistic regression, the per-sample loss function has two components: the logistic loss function and the regularization function. The logistic loss function ℓ_t (defined in (1)) has a convenient property that its gradient in dimension i is 0 if $x_{t,i} = 0$, thus allowing for sparse updates. Unfortunately, the regularization term $\frac{\lambda}{2}\sum_i(\beta_i - \mu_i)^2$ generally does not have zero gradients. In fact, the regularization

[4]Our algorithm uses different learning rates for different dimensions, rather than a global learning rate for all dimensions. Multiple studies suggest that dimension-specific learning rates yield superior performance to global learning rates, particularly for data sets with sparse features like ours [6, 9].

term is independent of x_t and has non-zero gradient in dimension i as long as $\beta_{t,i} \neq \mu_i$. This regularization term prevents us from running efficient sparse updates.

Our solution is to rewrite the per-sample loss function as the following:

$$\tilde{f}_t(\beta) = \ell_t(\beta) + \sum_{i:x_{t,i}\neq 0} \frac{\lambda}{2}\frac{N}{N_i}(\beta_i - \mu_i)^2 \tag{10}$$

here N is the total number of training samples, and N_i is the total number of times that dimension i is present in a sample. We essentially reassigned the regularization term to the per-sample loss function. Instead of assigning equal regularization strength to every data sample, we choose to only regularize a dimension when that dimension is present, and correspondingly increase the regularization strength of that dimension in inverse proportion to the number of samples that have this dimension. In so doing, we have kept the total regularization strength unchanged, i.e., $\frac{1}{N}\sum_{t=1}^{N} f_t(\beta) = \frac{1}{N}\sum_{t=1}^{N} \tilde{f}_t(\beta) = f$.

Our learning method operates on the rewritten per-sample loss function $\tilde{f}_t$ instead of f_t. At step t, for dimension i such that $x_{t,i} = 0$, since $\nabla_{t,i} = \frac{\partial \tilde{f}_t}{\partial \beta_i}|_{\beta=\beta_t} = 0$, we can safely ignore updates on these dimensions and only run updates (7) (8) (9) on dimensions i with $x_{t,i} \neq 0$.

Explicit Handling of Regularization.

The original NoPesky method doesn't explicitly handle regularization terms, effectively putting the regularization as part of the loss function. Hence the regularization term also contributes to the running estimates $g_{t,i}$, $v_{t,i}$ and $h_{t,i}$. This inhibits the integration of NoPesky with adaptive regularization—we would like it to be capable of handling changing regularization factors.

Fortunately, since the regularization term is deterministic and can be calculated precisely, we can exclude them from the calculation of the running estimates. Thus, instead of (8), we have

$$\begin{aligned} g_{t,i} &= (1-\tau_{t,i}^{-1})g_{t-1,i} + \tau_{t,i}^{-1} \left.\frac{\partial \ell_t}{\partial \beta_i}\right|_{\beta=\beta_t}, \\ v_{t,i} &= (1-\tau_{t,i}^{-1})v_{t-1,i} + \tau_{t,i}^{-1} \left(\left.\frac{\partial \ell_t}{\partial \beta_i}\right|_{\beta=\beta_t}\right)^2, \\ h_{t,i} &= (1-\tau_{t,i}^{-1})h_{t-1,i} + \tau_{t,i}^{-1} \left.\frac{\partial^2 \ell_t}{\partial \beta_i^2}\right|_{\beta=\beta_t}, \end{aligned} \tag{11}$$

We can then modify our three approximations to

$$\begin{aligned} E[\nabla_{t,i}] &\approx g_{t,i} + \lambda_{t,i}(\beta_{t,i} - \mu_i), \\ E[\nabla_{t,i}^2] &\approx v_{t,i} + 2\lambda_{t,i}(\beta_{t,i} - \mu_i)g_{t,i} + \lambda_{t,i}^2(\beta_{t,i} - \mu_i)^2, \\ E\left[\left.\frac{\partial^2 f_t}{\partial \beta_i^2}\right|_{\beta=\beta_t}\right] &\approx h_{t,i} + \lambda_{t,i}, \end{aligned}$$

here $\lambda_{t,i} = \lambda_t \frac{N}{N_i}$ is a time-sensitive dimension-specific regularization factor (λ_t comes from adaptive regularization below). This leads us to

$$\eta_{t,i} = \frac{1}{(h_{t,i} + \lambda_{t,i})} \frac{[g_{t,i} + \lambda_{t,i}(\beta_{t,i} - \mu_i)]^2}{[v_{t,i} + 2\lambda_{t,i}(\beta_{t,i} - \mu_i)g_{t,i} + \lambda_{t,i}^2(\beta_{t,i} - \mu_i)^2]}.$$

Algorithm 1 Hyper-parameter Free Learning

```
**Input**: $\tau_{\text{init}}$, $e$, $\alpha$, $\lambda_0$
**Data variables**: $\beta_i, \bar{g}_i, \bar{v}_i, \bar{h}_i \leftarrow 0$, $\tau_i \leftarrow 1$, $\lambda \leftarrow \lambda_0$
**for** $t = 1$ to $N$ **do**
    Draw a training sample
    **for** each dimension $i : x_{t,i} \neq 0$ **do**
        Calculate $g_i = \frac{\partial \ell_t}{\partial \beta_i}|_{\beta=\beta_t}$ and $h_i = \frac{\partial^2 \ell_t}{\partial \beta_i^2}|_{\beta=\beta_t}$
        $\bar{g}_i \leftarrow (1-\tau_i^{-1})\bar{g}_i + \tau_i^{-1} g_i$
        $\bar{v}_i \leftarrow (1-\tau_i^{-1})\bar{v}_i + \tau_i^{-1} g_i^2$
        $\bar{h}_i \leftarrow (1-\tau_i^{-1})\bar{h}_i + \tau_i^{-1} h_i$
        $g\text{reg} \leftarrow \lambda_t(N/N_i)(\beta_i - \mu_i)$
        **if** $\tau_i < \tau_{\text{init}}$ **then**
            $\beta_i \leftarrow \beta_i - e(g_i + g\text{reg})$
            $\tau_i \leftarrow \tau_i + 1$
        **else**
            ratio $\leftarrow (\bar{g}_i + g\text{reg})^2/(\bar{v}_i + 2\bar{g}_i g\text{reg} + g_{\text{reg}}^2)$
            $\eta_i \leftarrow$ ratio$/(\bar{h}_i + \lambda_t(N/N_i))$
            $\beta_i \leftarrow \beta_i - \eta_i(g_i + g\text{reg})$
            $\tau_i \leftarrow (1 - \text{ratio})\tau_i + 1$
        **end if**
    **end for**
    Draw a validation sample
    $\lambda_{t+1} = \lambda_t + \alpha\lambda_t \sum_{i=1}^{d} \frac{\partial \ell_t^{\text{val}}}{\partial \beta_i}|_{\beta=\beta_t} \beta_{t,i}$.
**end for**
```

Robust Slow-Start Initialization.

Finally, we address one more limitation to the original NoPesky method, when applied to our context. The NoPesky method relies on a number of approximations that get increasingly better as it runs more updates. Unfortunately these approximations can be inaccurate in the initial stages of the training when there are few updates and the coefficients are far from the optimal solution. The original paper [14] proposed a slow-start method to process the first 0.1% of the data with a small constant learning rate in order to obtain a stable estimate of the running averages. While this method works well in their setting, where data have dense rows, it doesn't work well with data having sparse rows because many dimensions might not show up at all in the first 0.1% of the data samples!

Thus, we introduce an alternative slow-start method. We use a small learning rate (ϵ) for updates *in a particular dimension* until we have seen at least τ_{init} samples having *that dimension.* Effectively we use a fixed learning rate SGD algorithm for the first τ_{init} updates in a dimension, and only after that do we start to engage the NoPesky method. We set $\tau_{\text{init}} = 100$ and $e = 10^{-6}$ and use this for all of our experiments [5].

4.2 Adaptive Regularization

A common strategy for selecting optimal regularization weights is to perform a grid search: train multiple models using different regularization factors, and select the one that yields the best performance on the validation data. As dis-

[5]While the slow start mechanism does in fact create hyper-parameters, we found that the same settings work for a large class of similar problems (i.e., all sparse campaign data sets) and no tuning is needed once an appropriate configuration is found for that class

cussed above, this is time-consuming and prohibitive when training very large numbers of models.

Adaptive regularization [13] offers an alternative solution. It adapts the regularization factor iteratively on the validation data while learning the model on the training data. Adaptive regularization works with L^2-regularized models and can be integrated into an SGD algorithm. Instead of training multiple models with different but fixed regularization factors, with adaptive regularization we train one model with regularization factors that change over time. This is achieved by running a gradient descent algorithm on the regularization weight.

Adaptive regularization works as follows: alternatively draw one sample from the training data and one sample from the validation data then run the SGD update for the training sample. Let ℓ_t^{val} be the logistic loss function of the validation sample. Adaptive regularization then runs the following update

$$\lambda_{t+1} = \max\{\lambda_t + \alpha \sum_{i=1}^{d} \left.\frac{\partial \ell_t^{\text{val}}}{\partial \beta_i}\right|_{\beta=\beta_t} \beta_{t,i}, 0\} \qquad (12)$$

where α is a fixed learning rate. The max is necessary because we would like to keep the regularization factor always a non-negative quantity.

Dimension-Homogeneous Update.

There are a few undesirable features of the update in (12). First, it is difficult to choose a learning rate α that works well across multiple magnitudes of the regularization factor. Second, the need to truncate at 0 in (12) seems artificial. Finally, the dimension of the update equation doesn't match, because the dimension of $\frac{\partial \ell}{\partial \beta}\beta$ equals the dimension of ℓ, which is a logistic loss function and hence dimensionless. But λ is not dimensionless and has dimension (dimension of $\beta^2)^{-1}$ based on (4).

We fix the above issues by slightly modifying the update equation to

$$\lambda_{t+1} = \lambda_t + \alpha\lambda_t \sum_{i=1}^{d} \left.\frac{\partial \ell_t^{\text{val}}}{\partial \beta_i}\right|_{\beta=\beta_t} \beta_{t,i}. \qquad (13)$$

By inserting λ_t to the gradient term, we restored the dimension homogeneity of the update formula. We found this modified update formula performs more robustly in practice.

We illustrate adaptive regularization at work in Figure 1. In this example, we take a one-day sample from our training data stream (the *source* data defined in the previous section) and perform several training runs, with each one having a different initial starting point for the regularization factor. The plot shows the path that λ takes as the optimization sees more data. We see that all paths converge to the same factor, though the speed to convergence is sensitive to the initial starting point.

In practice, we set the fixed learning rate $\alpha = 0.001$. Because we will perform many updates on the regularization factor, and the model changes slowly around the optimal regularization factor, the performance of our model is not sensitive to the choice of this learning rate.

5. EMPIRICAL ANALYSIS

In this section we demonstrate empirically the improvements conferred by transfer learning, as well as the robustness of the "hands-free" design. The analysis is conducted experimentally across 100 different campaigns, across a wide variety of advertisers from different industries, with different base rates, and with different inherent predictability. A quick summary goes as follows: 1) using the transfer learning from section 3 achieves better results than not using transfer learning, and 2). doing this "hands-free" via the methods detailed in section 4 does not perform much worse than doing a full-blown grid search that is (at least) an order of magnitude less efficient. In all of these experiments, we report results on experiments conducted on data from individual campaigns, either in full detail or in aggregate. In every case, the source and target data sets used are specific to that particular campaign (as described above).

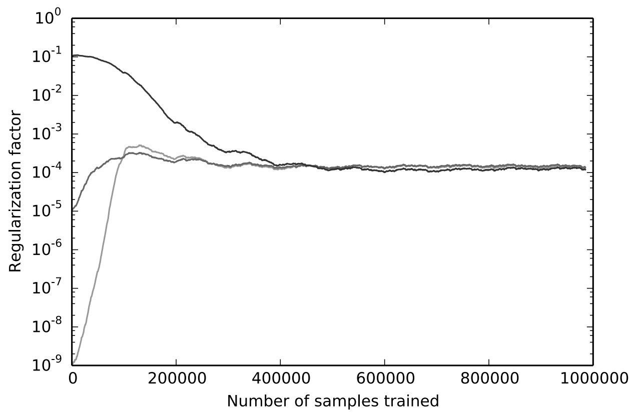
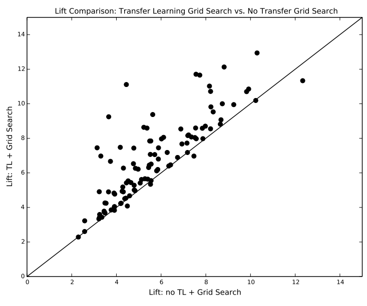


Figure 1: Adaptive regularization with dimension-homogeneous update formula (13) **converges to the same regularization factor starting from a wide range of initial values.**

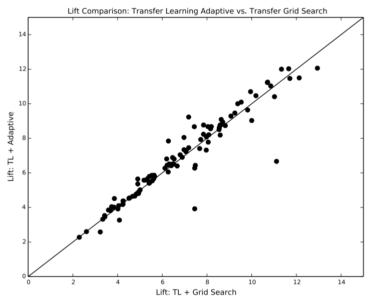
5.1 Transfer and Hands-free Operation

Our first set of experiments was conducted on data from a single time frame spanning two time periods. We chose 100 advertisers and for each advertiser created 3 data sets: (1) a source data set D_0^S in period τ_0, (2) a target data set D_0^T in period τ_0, and (3) a target data set D_1^T in period τ_1. In this scenario periods τ_0 and τ_1 are disjoint but consecutive, and τ_0 and τ_1 span fourteen and seven days, respectively.

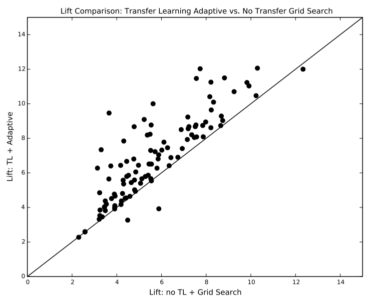
For this set of experiments we use a 2x2 factorial design for examining the modeling. One experimental factor is whether or not we use transfer learning. For the "Transfer Learning" variant we first learn a weight vector $\hat{\mu^s}$ using equation 2 on the source data D_0^S. We then use D_0^T, $\hat{\mu^s}$ and equation 3 to learn a source-domain-adapted model on the target data. For the "No Transfer" (control) variant, we again use D_0^T and equation 3 but without the informative prior $\hat{\mu^s}$. The second design factor represents how we tune the hyper-parameters when training the models. For the "Adaptive" variant, we use the adaptive regularization with the NoPesky learning rate schedule. For the "Grid Search" variant we use the AdaGrad learning rate formula and specifiy a fixed regularization weight. We search over a 4x9 grid of values, where we use all powers of ten between 10^0 and 10^{-3} for the initial AdaGrad learning rate and 10^2 to 10^{-6} for the regularization weight. In all training scenarios we



(a)



(b)



(c)

Figure 2: These plots show experimental results on 100 campaigns under different learning scenarios. Each point within each plot shows the lift evaluated on the same holdout set using the two methodologies listed. Plot (a) isolates the effect of using transfer learning vs. not using transfer learning using traditional grid-based hyper-parameter search for regularization and AdaGrad learning rates for both. We see that transfer learning almost always improves performance on the target data set. Plot (b) compares transfer learning using grid search vs. the hands-free methods we introduce in this paper. We see that hands-free is competitive with the more expensive approach. Plot (c) shows our main result, comparing transfer learning with adaptive learning vs. no transfer learning and traditional grid search. This combines methods from (a) and (b), and demonstrates the net performance effect from the integrated, enhanced method.

select hyper-parameters by splitting the training data into training and validation sets. We choose the hyper-parameter configuration that performs best on the validation set and then retrain on the entire training set using the selected hyper-parameters.

Figure 2 shows[6] the results of multiple tests on the different training scenarios. In figure 2 (a) we first establish that transfer learning improves performance under traditional hyper-parameter selection methods. In both axes, the models were trained using our Grid Search variant and the only difference is that transfer learning was used to produce the results in the vertical axis. Overall, we see that 91% of points fall above the identity line, meaning transfer learning improves performance for an overwhelming majority of cases.

In figure 2 (b) we hold the transfer learning variant constant and test whether our adaptive hyper-parameter selection strategies fare better or worse than traditional grid search. Overall we see a balanced and tight scatter around the identity line, with the exception of two campaigns. This suggests that in the context of our transfer learning strategy, adaptive hyper-parameter search is as effective as more traditional grid search methods. This is a powerful result, as it suggests we can indeed have a "hands-free" approach to training our models while keeping the training costs low. Our grid search optimization required 36 runs through the training data whereas the adaptive approach only takes one. In a more realistic setting, we may be able to reduce the grid space, but even then, we'd expect to still require 5-10 training runs to find an optimum. Thus, the adaptive approach significantly speeds up our ability to build transfer-enhanced models.

In figure 2 (c) we show the variants of transfer learning with adaptive hyper-parameter search against no transfer with grid search. This is our closest comparison between our proposed approach and what might be done without access to the source data. We see that 2 (a) and 2 (c) are qualitatively similar, with the difference being that (c) can be achieved up to 10 times faster because of the adaptive learning.

[6]Note that all 3 plots in figure 2 evaluate generalization performance using lift. Using AUC instead shows the same results qualitatively. See Table 1 for summary results.

The results of these three comparisons are summarized in table 1. In each row we report the mean of differences when training according to the strategies specified in the first column. For AUC we measure the difference between the two strategies and for Lift we measure the ratio between the two. In our experimental design, we set our null hypothesis to be that using one variant produces no difference in performance over the other variant. We report the mean AUC difference and Lift ratio, in the 2nd and 5th columns, respectively (we take $1^{st} - 2^{nd}$ for AUC and $1^{st}/2^{nd}$ for lift). We also report the p-Values using a paired t-test on these statistics. If we think about these methods as global strategies for all campaigns, we can conclude with strong confidence that transfer learning improves campaigns on average and adaptive learning does not hurt.

5.2 Transfer and Incremental Training

In this set of experiments we explore a scenario that reflects an additional aspect of production training and model use at a firm like Dstillery. In particular, data arrives incrementally and continually from the moment a new campaign is initiated until it runs its course, and (hopefully) is replaced by a new campaign.

Since SGD trains via incremental updates, it can run naturally as an incremental learning system. For a given campaign, at any point in time we have a targeting model that has been trained using all data available up until that point; this model will then be used to target the following time period's ads. We simulate this process by creating daily samples from the advertising system for each campaign, and then incrementally updating the campaign's model using some experimental learning-algorithm variant.

	AUC Diff. Mean	AUC Diff. % > 0	AUC Diff. p-Value	Lift Ratio Mean	Lift Ratio % >1	Lift Ratio p-Value
Transfer:Grid vs. No Transfer: Grid	2.07	86%	$< 10e^{-6}$	1.22	91%	$< 10e^{-6}$
Transfer:Adapt vs. Transfer: Grid	-0.30	50%	0.15	1.01	65%	0.09
Transfer:Adapt vs. No Transfer: Grid	1.77	80%	$< 10e^{-6}$	1.23	93%	$< 10e^{-6}$

Table 1: Each row in this table is a comparison of two methods on the 100 campaigns shown in figure 2. The table reports the mean difference of AUC (on a 100 point scale) between the first and second variants (listed in the first column) and the mean ratio of Lift for the first variant over the second variant across all the campaigns. The p-Values are a result of paired t-tests under the null hypothesis that the mean AUC difference = 0 and mean Lift ratio = 1. The columns labeled '%>k' show the number of campaigns where the metric reported was strictly greater than k. These results show that transfer learning overall improves the performance (over not using it) and that the hands-free learning does not substantially reduce predictive performance.

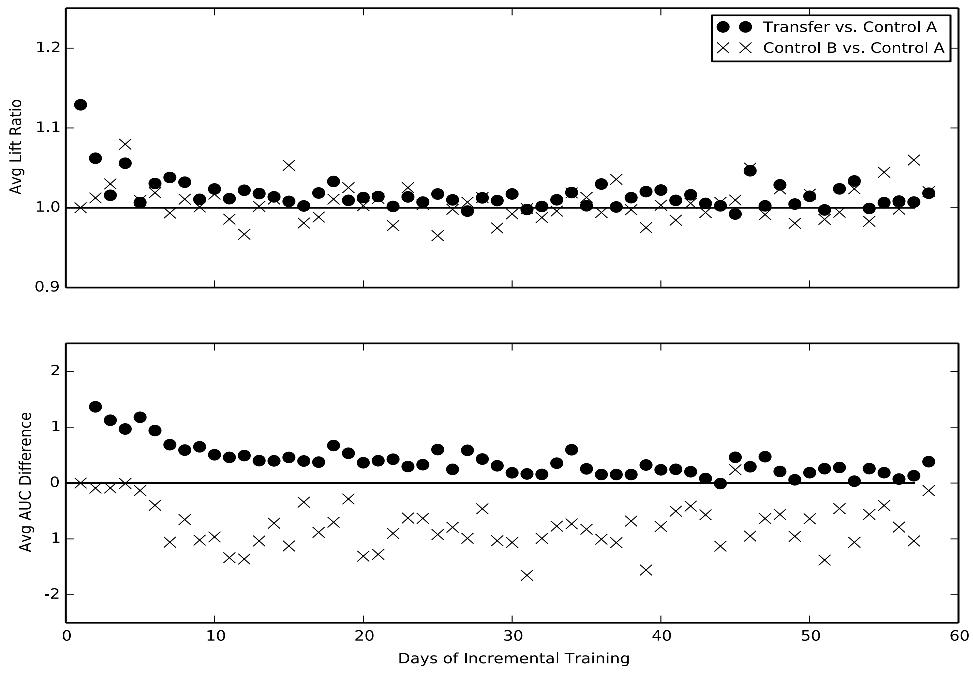


Figure 3: This figure shows the comparative results for three strategies for incremental training. For each day of analysis we initialize a model from the prior day, train on campaign data for the specified day and evaluate on campaign data for the following day. We explore 3 modeling strategies (Transfer Learning, Control A with no prior and Control B with previous day's model as a prior). Each point compares either Transfer Learning or Control B variants to the Control A variant.

We then evaluate the resultant model on the following day's sample.

We explore three variants in this set of experiments: (1) "Transfer Learning" – using equation 3 with $\hat{\mu}^s$ derived from the source data;[7] (2) "Control A" – equation 3 with $\hat{\mu}^s = 0$ (the standard regularization case), and (3) "Control B" – equation 3 with $\hat{\mu}^s$ specified as the prior day's model. This final "Control B" variant is the methodology used by [4], and is the method most similar to our work in design.[8] For each variant, on each day, we train on just the given day's data using the chosen prior with adaptive regularization and optimize using SGD with the NoPesky adaptive learning rates. When performing incremental updates using SGD, we found that performance generally improves when each day's model is initialized with the prior day's model and that the state of all parameters used to compute the learning rate are persisted and carried over to the next day (let's call this the 'warm start').

[7]In this set of experiments we learn one estimate of $\hat{\mu}^s$ for each campaign and hold that fixed throughout the 60 days of training.

[8]The work by [4] was not explicitly attempting knowledge transfer, but can be interpreted as an instance of Bayesian transfer, like ours except transferring across time periods.

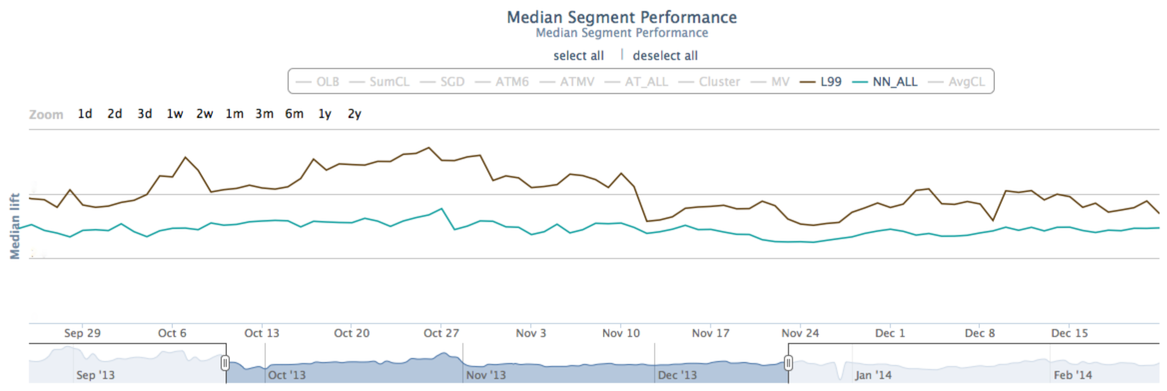


Figure 4: Snapshot of the Dstillery KPI dashboard showing median lift over an internal baseline across all active campaigns. The 'L99' line is the median Lift KPI of our transfer learning algorithm across all campaigns. The 'NN-ALL' line is the median Lift KPI across all campaigns of all of our learning algorithms combined.

Figure 3 shows aggregated results of running incremental training with daily performance evaluation across 30 randomly sampled campaigns from our production system (sampled with the one constraint that the target data had at least 5 post-impression conversions for each day of analysis). In the top chart we show the average relative Lift ratio between the variants Transfer Learning and Control B over Control A. In the bottom chart we show the average difference in AUC. Thus, in each chart a black dot above the reference line means that Transfer Learning outperformed Control A for that time period. An "X" above the line means that Control B outperformed Control A for that time period. A black dot above an "X" means that Transfer Learning outperformed Control B for that time period. And vice versa. From these results we can see multiple effects: (1) when comparing to Control A, the postive effect of transfer learning wears off over time; (2) when comparing to Control A, the transfer learning benefit is more pronounced for AUC than for Lift, although for both metrics there is a general benefit, and (3) Control B tends to underperform both variants, and this is more dramatic in AUC.

The first trend mentioned above is something we ought to expect - as more data is introduced into the system, and because the models do have some form of a memory built into them, we should expect the weights learned just with the target data will start to converge toward an optimal point. The second trend can possibly be explained by different properties of the metrics themselves. In general, Lift can be a very high variance metric in data scenarios with a low absolute number of positive outcomes. This variance comes from both (i) the model, because it is more difficult to fit the tails of the distribution, and (ii) the evaluation itself: since we are updating the models one day at a time, we have less target data per model-building and evaluation step and this increases the expected variance. As a policy, we always review both AUC and Lift, even though Lift is the more appropriate metric.

The third trend mentioned above might be the most unexpected (to us at least), and the sub-optimality of Control B warranted additional analysis. Figure 5 shows the average L2-norm of the weight vectors β learned for each variant for each day. The general trend we see is that the average

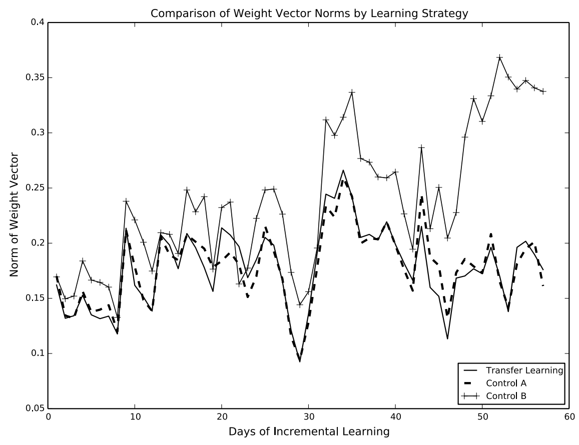


Figure 5: Average L2-norms for the models aggregated and reported in figure 3

norm of weight vectors trained using the Control B variant is bigger than that of the other two. This trend is likely an artifact of regularizing towards existing models. The issue is two-fold and manifests from the fact that the regularization prior is a moving target. Over time, the weight vector gets closer to an optimal solution and this often coincides with an increasing norm. Eventually the process begins to overfit the data and the regularization is too restrictive to let the model escape from a specific solution. Also, this regularization strategy makes it difficult to escape optimization valleys in non-stationary data. The NoPesky learning rate schedule has a built-in method for adapting to changing data distributions (i.e., by "forgetting" most of the past), but forgetting the past is difficult when the regularizer forces the model to remember it. Thus, the Transfer Learning prior might be better in our application because it doesn't increase over time and is biased enough to avoid overfitting. It is anchored close enough to a specific day's optimal solution to be useful but not too close that it over-fits in a generally noisy learning environment.

5.3 Production Performance

As additional, auxiliary support that the hands-free transfer method works well, we present its actual production performance. In the experiments above, we took care to control for the different factors that affect performance on a holdout set so that we could isolate the different effects of our design choices. Once released 'into the wild' there are many elements that prevent a pure apples-to-apples comparison (when not doing a pure A/B test on them). Nonetheless, we find in particular that our implementation of hands-free transfer learning is in aggregate our best-performing learning method.

Figure 4 shows a screen shot taken from our internal KPI dashboard. We benchmark every targeting algorithm for every campaign against a default baseline that runs as part of the campaign. We normalize the conversion rates of each algorithm in each campaign against its respective baseline and call this our "Lift KPI." Figure 4 here shows the median Lift KPI across all campaigns for the fourth quarter of 2013. The top line (called 'L99'[9]) represents our transfer learning algorithms in action. The bottom line (called 'NN-ALL') is a blend of all our algorithms combined. While this 'L99' group might not perform best for every campaign, in general it is our best and demonstrably performs well above average across all campaigns. Many of the fluctuations in relative performance between these two lines are driven by the internal campaign allocation optimizations of the system. Each campaign starts with a set of competing algorithms. As we get live feedback we reallocate impressions to the better performing algorithms until the algorithm's performance converges to the mean of the campaign. Thus, as with production numbers generally, these should not be taken as an exact comparison between algorithms under identical conditions.

6. CONCLUSION AND DISCUSSION

We have presented a set of techniques that when applied together present a robust and scalable solution to several challenges often encountered in online display advertising. Our transfer learning approach shares intuition with prior work [1], [8], in that one can look outside of the target data set for sources of signal that can be passed to the target problem. An advantage to our formulation is that (in retrospect) it is very straightforward, which is a virtue when designing and implementing large-scale, completely automated machine learning systems. Our focus here is on producing a scalable, accurate, and robust system. We have achieved that through the combination of Bayesian transfer learning and the extension and integration of methods for adaptive, parameter-free learning. Looking forward, adaptive regularization is a recently developed strategy and more research needs to be done to understand its performance bounds and general limitations. On the transfer learning side, the effectiveness of the strategy depends on specifying an appropriate prior. This work presents a straightforward mechanism for how to transfer; more sophisticated methods could increase the advantages from inductive transfer. Additionally, within this framework, understanding *when* to transfer is necessary for developing a fully robust solution.

[9]We often use obscure terminology to refer to our internal products

7. REFERENCES

[1] D. Agarwal, R. Agrawal, R. Khanna, and N. Kota. Estimating rates of rare events with multiple hierarchies through scalable log-linear models. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 213–222. ACM, 2010.

[2] A. Arnold, R. Nallapati, and W. W. Cohen. Exploiting feature hierarchy for transfer learning in named entity recognition. In *ACL*, pages 245–253, 2008.

[3] M. Broadie, A. Zeevi, and D. Cicek. Multidimensional Stochastic Approximation: Adaptive Algorithms and Applications. 2013.

[4] O. Chapelle, E. Manavoglu, and R. Rosales. Simple and scalable response prediction for display advertising. *Transactions on Intelligent Systems and Technology*, 2013.

[5] C. Chelba and A. Acero. Adaptation of maximum entropy capitalizer: Little data can help a lot. *Computer Speech & Language*, 20(4):382–399, 2006.

[6] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, pages 2121–2159, 2011.

[7] Interactive Advertising Bureau. Q3 2013 earnings press release. `http://www.iab.net`.

[8] Y. Liu, S. Pandey, D. Agarwal, and V. Josifovski. Finding the right consumer: optimizing for conversion in display advertising campaigns. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 473–482. ACM, 2012.

[9] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, S. Chikkerur, D. Liu, M. Wattenberg, A. M. Hrafnkelsson, T. Boulos, and J. Kubica. Ad click prediction: a view from the trenches. In *KDD '13: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1222–1230, New York, New York, USA, Aug. 2013. ACM Request Permissions.

[10] S. J. Pan and Q. Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010.

[11] C. Perlich, B. Dalessandro, T. Raeder, O. Stitelman, and F. Provost. Machine learning for targeted display advertising: Transfer learning in action. *Machine Learning*, pages 1–25, 2013.

[12] R. Raina, A. Y. Ng, and D. Koller. Constructing informative priors using transfer learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 713–720. ACM, 2006.

[13] S. Rendle. Learning recommender systems with adaptive regularization. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 133–142. ACM, 2012.

[14] T. Schaul, S. Zhang, and Y. LeCun. No more pesky learning rates. *arXiv preprint arXiv:1206.1106*, 2012.

[15] W. Xu. Towards optimal one pass large scale learning with averaged stochastic gradient descent. *arXiv preprint arXiv:1107.2490*, 2011.

[16] M. D. Zeiler. ADADELTA: An Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701*, 2012.