Notification Volume Control and Optimization System at Pinterest

Bo Zhao, Koichiro Narita, Burkay Orten, John Egan
Pinterest
San Francisco, CA
{bozhao,koichiro,borten,jwegan}@pinterest.com

ABSTRACT

Notifications (including emails, mobile / desktop push notifications, SMS, etc.) are very effective channels for online services to engage with users and drive user engagement metrics and other business metrics. One of the most important and challenging problems in a production notification system is to decide the right frequency for each user. In this paper, we propose a novel machine learning approach to decide notification volume for each user such that long term user engagement is optimized. We will also discuss a few practical issues and design choices we have made. The new system has been deployed to production at Pinterest in mid 2017 and significantly reduced notification volume and improved CTR of notifications and site engagement metrics compared with the previous machine learning approach.

CCS CONCEPTS

• **Information systems** → **Information systems applications**;

KEYWORDS

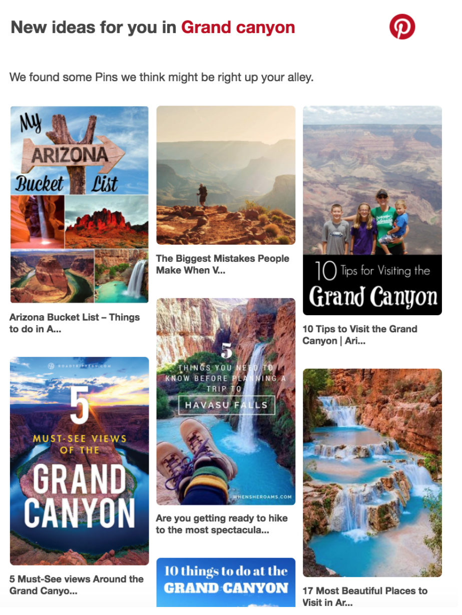
Notification Volume Optimization; Machine Learning

ACM Reference Format:
Bo Zhao, Koichiro Narita, Burkay Orten, John Egan. 2018. Notification Volume Control and Optimization System at Pinterest. In *KDD '18: The 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, August 19–23, 2018, London, United Kingdom.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3219819.3219906

1 INTRODUCTION

Pinterest is one of the world's largest search and recommendation engines [6, 10, 12]. There are more than 200 million users who come to Pinterest every month to search and discover pins that match their interests. Majority of users come to the site directly either from their browsers or mobile apps, in the meantime, we leverage additional channels to reach and engage users, for example, allowing search engines like Google to index our content and surface them in search results, encouraging users to share content with their friends in other social networks, and most importantly, sending relevant content to users via notifications. At Pinterest, billions of notifications are sent every week to our users and drive a significant portion of monthly active users.

KDD '18, August 19–23, 2018, London, United Kingdom
© 2018 Association for Computing Machinery.
ACM ISBN 978-1-4503-5552-0/18/08…$15.00
https://doi.org/10.1145/3219819.3219906


New ideas for you in Grand canyon

We found some Pins we think might be right up your alley.
My ARIZONA Bucket List
Arizona Bucket List – Things to do in A...
The Biggest Mistakes People Make When V...
10 Tips for Visiting the Grand Canyon
10 Tips to Visit the Grand Canyon | Ari...
5 Things you need to know before planning a trip to Havasu Falls
Are you getting ready to hike to the most spectacula...
5 Must-See Views of the Grand Canyon
5 Must-See views Around the Grand Canyo...
10 things to do at the GRAND CANYON
17 Most Beautiful Places to Visit in Ar...

Figure 1: Sample pin recommendation email on Grand Canyon

There are multiple channels to send notifications, including emails, mobile push notifications, desktop push notifications, SMS, *etc.* If the content is relevant and the timing is right, notifications could serve as a natural extension of the core product and provide users with great experience. For example, emails allow casual users to view recommendation content directly in their email clients without logging into the site or installing the mobile app. Push notifications are also very effective to engage with more active users who have the Pinterest app on their phones. Moreover, we find a clear concise subject line that summarizes the recommendation content is very important to improve the user experience for notifications, since it allows users to easily decide if they would like to open or ignore the emails or push notifications. Figure 1 is an example of pin recommendation emails about a specific topic, users can view these scenic pictures and click the ones that they want to save or know more details about, and in this case "Grand Canyon" will be mentioned in the email or push subject line, so users can make a quick decision on if they are interested in the topic. Besides pin recommendations, we also have board recommendations, interest recommendations and a dozen of other recommendation types, and advanced machine learning models have been developed to personalize these types and content for each user to improve relevance and user experience.

While content relevance is clearly the cornerstone for improving user experience with notifications, there are many other problems to consider in a practical notification system: when to send, which channel to send to, and most importantly, how frequently to send notifications to each user. Indeed, in practice notification volume is the most important lever to control. Obviously, simply increasing volume to every user could dramatically boost site engagement metrics in the short term, however, if the volume is too high, it could result in several negative consequences: first, users could unsubscribe the emails or uninstall the mobile app so that we could not send emails or push notifications to them anymore; second, it could increase the chance that email service providers and mobile operating systems treat the site as a potential spammer and start to block its messages; and finally, it could hurt users' trust in the brand. All of these consequences could negate the short term gain and actually cause a long term loss on engagement metrics. Therefore, it is a very important problem to develop intelligent algorithms to optimize notification volume for each user, which is the focus of this paper.

There are several challenging issues to address in the notification volume optimization problem. First, it is important to have a proper objective function that captures the long term utility of sending notifications instead of only optimizing for short term effect. Second, increasing volume typically has a diminishing return on the utility function, so it is important that the framework is able to leverage nonlinear models to capture this effect. Third, the optimization algorithm needs to be very efficient and scalable to handle hundreds of millions of users. Last but not the least, there are many practical issues to consider, *i.e.,* how to handle multiple notification channels including emails and push notifications, how to manage the interaction between the volume control component and content ranking components such that new notification types and ranking models can be easily developed and tested, *etc.*

In the following sections of the paper, we will introduce the notification volume control and optimization system we recently developed at Pinterest, which provides a feasible solution to the challenges mentioned above. The new system has been fully deployed to production in mid 2017, and compared with the previous system which is also machine learning based, we significantly reduced notification volume and improved CTR of notifications and site engagement metrics. We will discuss our novel objective function and machine learning approach, as well as some of the practical design choices we have made and the reasoning behind them. In the related work section, we will also discuss some previously published systems [7, 8], and explain the difference and advantages of our proposed approach.

2 RELATED WORK

There are mainly three areas of work related to the topics covered in this paper: including spam classification and detection, email action prediction and prioritization, and some recently published methods on email volume optimization.

Spam detection is a well studied area and there are numerous papers on this topic. In most cases, email service providers develop spam detection methods to filter spam from the inbox and improve the experience of their users. For example, [11] described a method to score sender reputation for antispam in Gmail. [3] developed a hierarchical Bayesian method that models the common distributions for groups of users. Spam detection is related to our work since as notification senders, we also build similar machine learning models to predict users' negative actions such as unsubscribing or labeling the messages as spam, and leverage such prediction to adjust volume accordingly: intuitively if the user has higher probability to unsubscribe we will need to reduce the volume.

While spam detection focuses on negative actions, there are more recent work that aim to predict a wide range of positive actions, including open, reply, forward, *etc.* [1] introduced the model that was developed for Gmail priority inbox. A logistic regression model is trained, using features including social features, content features, *etc.* A simple transfer learning scheme is also leveraged to train a global model and per user models. Yahoo [4] also developed a framework to predict various actions of emails. The model leverages local features for individual inboxes, and global features to capture the overall mail traffic. The framework trains vertical models for personalization over a single inbox and horizontal models for regularization and capturing behaviors of similar users. Predicting positive actions of notifications is relevant to our work because as senders we also build models to predict click through rate for each user on notifications and leverage them to decide the volume: in general, if the user likes to interact with emails or push notifications, we probably can send them a bit more frequently.

More recently, LinkedIn proposed the problem of email volume optimization for large scale online services [7, 8]. This is most related to our work in this paper. More specifically, [8] proposed to leverage multi-objective optimization by using separate constraints to set lower bounds of positive actions (*e.g.,* downstream sessions) and upper bounds of negative actions (*e.g.,* unsubscribe) for volume distribution plans. The optimization is based on methods proposed in [2]. [7] is an extension of this framework by changing the positive actions from downstream sessions to active users, such that the volume plans are more effectively optimized for site-wide engagement metrics. As we will explain in details in later sections, there are a few limitations of this approach, which we aim to improve in our new framework. First, positive actions and negative actions were treated as separate constraints, and there are global parameters that need to be tuned to control the total number of positive and negative actions. In our framework, we propose a novel method to model the long term effect of notifications for each individual user, such that the positive and negative actions are tightly integrated and their weights are automatically learned for each user. Second, there are some simplified assumptions made in [7, 8], namely, the effect of each email send is treated independently and the total effect of k emails is simply the sum of individual emails. However, in practice notifications are not independent and there is a clear diminishing return of sending more to each user. Moreover, in [7] the prediction model for site engagement is also limited to be a linear model for the optimization framework to work. In our new framework, we are able to fully leverage the power of non-linear models such as boosted decision trees to capture the nonlinearity

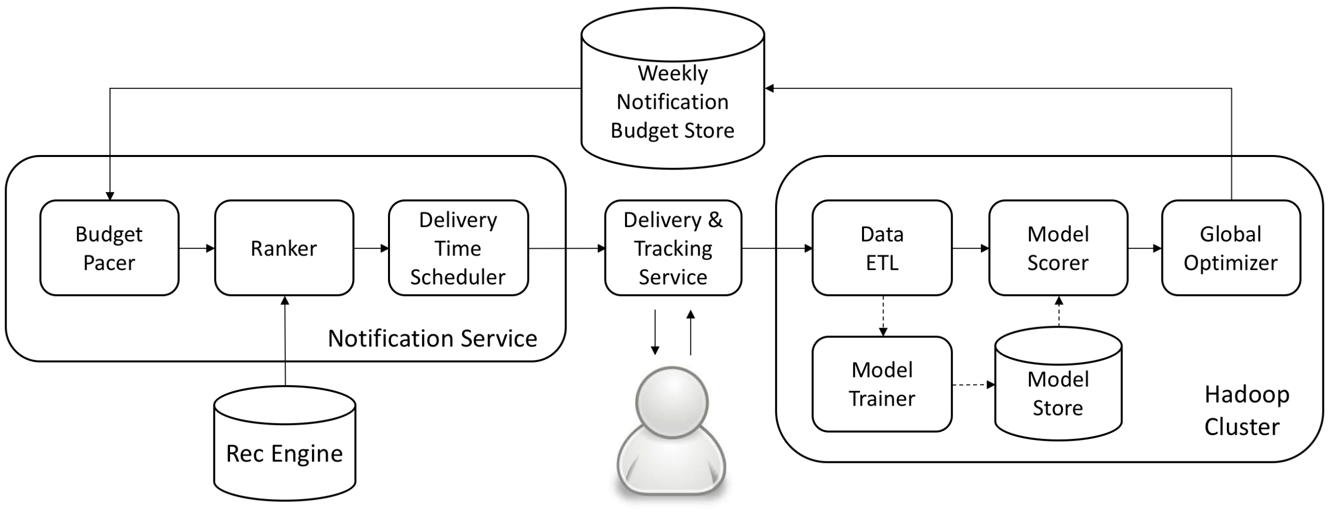


Figure 2: Pinterest notification system diagram

in the data and improve prediction accuracy. Last but not the least, there are a few practical advantages of our new framework for a large scale online service and an active engineering organization where many parts of the notification system are being built and tested frequently. Specifically, [7, 8] requires a quadratic programming solver, which is computationally expensive for processing hundreds of millions of users, while our method is much more efficient. Another important point is that in [7, 8], CTR models for various email types are tightly integrated with the volume control system. However, we have learned practical lessons and made a cautious design choice to decouple the volume control component and type-specific CTR prediction and ranking components, in order to experiment with new notification types and ranking model improvements more easily and rigorously. Our framework also uses a unified model to capture the joint effect of multi-channel notifications including push notifications, which become more and more important in the mobile internet era.

3 SYSTEM OVERVIEW

In this section, we will briefly describe the overall system architecture of the notification delivery system at Pinterest, with the focus on the components of the volume optimization subsystem, and how it works together with other main components of the entire system. We will discuss several design choices we have made and how we reached these decisions from the lessons we learned from the previous volume control mechanism.

Figure 2 is a brief diagram of the main components of the notification delivery system at Pinterest.

3.1 Weekly Notification Budget

First of all, the most importance concept in our system is that we compute a weekly notification "budget" for each user, which controls the maximum number of notifications each user can get in a week. Most notifications we send out at Pinterest, which are content recommendations, are enforced by this budget, with a few exceptions including notifications for user messages, *etc.* We would like to point out that this is one of the main differences between our system and the previous volume control mechanism we developed and previously published methods [7, 8]. We will discuss the advantages of this design in more details in later of the section after we briefly describe all the main components.

3.2 Notification Service

Each day, the notification service processes each user and tries to decide if the user is eligible to receive notifications, and generates the content of the notification. It has several main components as follows:

Budget Pacer The weekly notification budget for each user is computed and stored in an online key-value store with user id as the key, and the role of the budget pacer is to fetch the weekly notification budget for each user and schedule it to each day, and the user is only eligible to receive notifications if there is budget on that day. The simplest pacing algorithm is *even pacing*, specifically, it tries to spread notifications as far as possible from each other, trying to minimize user fatigue. For example, if the weekly budget is 3, one possible send plan from even pacing would be Monday, Wednesday, and Saturday. Same logic applies if the budget is above 7 for some users: it would send one notification every day and evenly distribute the remaining ones. Note that the budget pacer also makes decisions daily based on the actual send history (or spent budget) of the week, for example, if the weekly budget is 1 and the original plan is to send on Wednesday but for some reasons (*e.g.,* no content) the notification is not sent, the pacer will try to send it in the remaining days of the week. More intelligent day of week optimization can be developed to improve user engagement, *e.g.,* trying to send more on weekends or based on actual user behaviors.

Ranker After the pacer decides the user is eligible to receive notifications, the ranker will rank from a list of notification types and select the best one to send. Machine learning models are developed to predict the CTR of notifications and other user behaviors, leveraging a comprehensive set of features including user engagement history, last time sent of the same notification type, *etc.* The model also considers notification channels, *e.g.,* emails or push notifications. Majority of notification types we send at Pinterest are content recommendations, including pin and board recommendations for particular topics, interest recommendation, user follow recommendation, *etc.* For each notification type, its content is generated based on specific recommendation algorithms, where machine learning models are used to select the most personalized content for each user. We will not go into details for this part since it is out of the scope of this paper.

Delivery After the notification content is generated, the delivery time scheduler will schedule at the time of the day when the user is most likely to engage. And the delivery service will send the notification at the scheduled time to the user. Note that the user could have multiple notification channels available, including email, mobile push, *etc.* The simplest strategy is to send via all available channels, and another strategy is to follow a certain order of channels to send. After delivery, the tracking service is able to track the user's response to notifications, which is very important to collect data to train various machine learning models for content ranking and volume optimization.

3.3 Volume Optimization Workflows

Volume optimization is performed by several offline data workflows that run on our Hadoop clusters. Data ETL jobs take the input of notification tracking data, and many other data sources (*e.g.,* site activity tracking data, user profile data) and compute features and labels used for training and scoring machine learning models.

We will discuss the specific machine learning models in later sections of the paper, and only give a high level overview here. Periodically, the model training workflow will train several models used in volume optimization and write them in a model store. Model scoring workflows score users and generate predictions base on each model. And an efficient global optimizer takes the prediction scores of all users and a global volume constraint to compute the weekly notification budget for each user. The budget data will be uploaded to the key-value store for online serving.

3.4 Design Choices

Now we have briefly introduced the key components of the notification system, we can discuss the reasoning behind some of the design choices we have made in details, including the practical lessons we learned from our previous system.

Legacy Volume Optimization Mechanism It is important to describe our previous system for volume optimization so we can better explain the key differences and advantages of the new system. Our previous system is also based on machine learning models and at high level shares some similarities with the method described in [8] although it is more simplified. The general idea of the method is very intuitive: if the user has higher probability to interact with certain types of notifications, she or he should receive those types more frequently. A set of global frequency rules are associated with the CTR prediction percentile of various notification types for each user and the overall user activity level. For example, if an active user's predicted CTR for notification type A is among top 10% of all users, s/he is allowed to receive A at most every three days, meaning the same notification type can only be sent two days after the previous send; and if a casual user's CTR for type B is among bottom 20% of all users, s/he is allowed to receive B at most every 14 days, *etc.* Everyday, the system will first decide what are the eligible notification types for each user and send out the one with the highest CTR prediction. These frequency intervals are hand tuned based on A/B experiments.

As we can see, in our previous system, similarly in the system described in [7, 8], frequency parameters are either hand tuned or automatically optimized at the notification type level, but there is no overall control of the total number of notifications each user can get in a period of time like a week. And the frequencies are tightly coupled with the CTR predictions of notification types. In practice, we have found this mechanism has several issues. First, in an active engineering organization like Pinterest, many teams are developing new notification types and want to easily test them in production A/B experiments. However, if we simply add the new notification type, train a CTR model, and reuse the same global frequency rules, it will result in significant notification volume increase, which will likely have gains on engagement metrics but it is impossible to understand whether the gain comes from simply increasing the volume or the quality of the new type. On the other hand, it is very difficult to tune all the frequency parameters such that the total volume remains the same in control and experiment groups, and even if we are able to do so, too many things have changed and the effect of the new type is still unclear. The exact same situation happens when we improve the type CTR prediction models or simply retrain them, since the volume mechanism is tightly coupled with those models, typically the new models will cause significant volume changes, making it very difficult to judge if the accuracy of the new models improves from A/B experiments.

Due to the above lessons we learned from the previous system, the first order design requirement we envision for the new system is to have an overall control of total notification volume for each user, and decouple the volume control component from the type ranking component. This design allows us to easily experiment with new notification types and ranking model improvements without worrying about significant volume changes, and therefore the experimental results and launch decisions are much more rigorous. Another practical benefit of the new system is that it can easily support custom rules and user settings on notification frequency (*e.g.,* at most once a week). Monitoring, analytics and diagnostics of overall notification system health are also easier since week over week total volume is much more stable and predictable.

From the modeling perspective, it is also more reasonable to directly optimize for the total number of notifications each user gets in a period of time. One key assumption made in previous work [7, 8] and our previous system is that each notification send is independent, and the total effect of multiple notifications is simply the sum of the effect of each individual ones. However, this assumption is too simplistic, since intuitively increasing volume has a diminishing return on user activity level due to user fatigue. In our framework, we do not make such independent assumption and we aim to directly estimate the effect of multiple notifications, which we will describe in details in later sections.

4 PROBLEM FORMULATION

In this section, we will explain how we formulate the problem of notification volume optimization and describe the objective function to optimize in details. High level speaking, the volume optimization problem can be treated as a constrained optimization problem: with a given total number of notifications, we try to figure out the optimal distribution among users such that certain objective function is maximized. There are two key factors to consider in choosing the right objective function. First, what is the target business metric we would like to improve, and what is the relation between the target metric and notification volume. Second, how to model the long term effect of notifications towards the target metric, which should consider the possibilities of both positive and negative actions from the user.

4.1 Utility of Notifications towards Target Business Metric

It is very important to choose the right target business metric to improve with notifications. In our previous system, we send more notifications to users who have higher CTR on notifications. Essentially, this means notification engagement metric is the implicit target metric to optimize in our previous system. Although this is quite reasonable, it may not be exactly what we want. In most companies, the more important metric we want to improve is the overall site engagement metric, such as daily active users (DAU), monthly active users (MAU), *etc.* Is the overall site engagement metric correlated with notification engagement metric? Yes. Will improving notification engagement cause site engagement metric to improve? Not necessarily, and we will explain the reason in details.

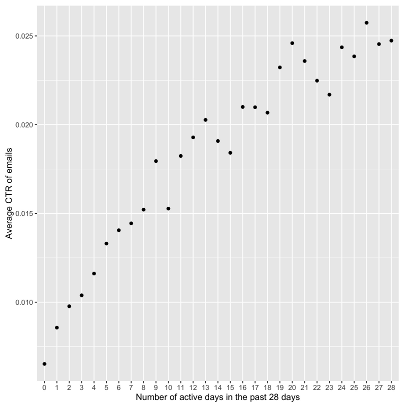


Figure 3: Email CTR v.s. user activity level

Figure 3 plots the average CTR of an email type among users at different activity levels (*i.e.,* number of days they are active in the past 28 days). The correlation is clear: more active users generally have higher CTR. However, this is because more active users are more likely to interact with all parts of the site, including notifications. To better illustrate this point, let us make some simplified assumptions on how users come to the site. We know that users could either come to the site organically, or they receive a notification and come to the site by clicking the notification. If we assume these two channels are independent, we can write the probability of a user being active as:

$$p(a|u) = p(a_o|u) + \underbrace{(1 - p(a_o|u)) \times p(a_n|u)}_{\text{Utility of notifications}}, \quad (1)$$

where $p(a_o|u)$ is the probability that the user comes to the site organically, and $p(a_n|u)$ is the user's CTR of notifications. As we can see, the second term of the summation is the incremental contribution (or utility) of notifications towards the user's overall activity level. This shows that if we want to directly optimize for overall site engagement, we should send more notifications to users with the highest utility scores instead of the highest notification CTR. Figure 4 plots this simplified utility score w.r.t. user activity level. The utility is highest for casual users who like to use the service sometimes but not very often. If the user is already very active, the incremental value of sending s/he notifications is lower, and the same is true for users who are very inactive.

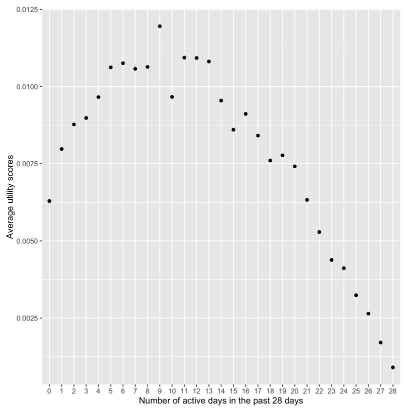


Figure 4: Simplified notification utility v.s. user activity level

Now we have explained the concept of the utility of notifications, we want to point out that Equation 1 is based on a simplistic assumption that various user visitation channels are independent with each other, which is not true in practice. Therefore, it is not exactly how we model the utility of notifications in this work, but only used to illustrate the point. In our system, we aim to directly model the complex interactions between notification volume and site activity, specifically, we try to estimate the conditional probability of a user u being active given a weekly notification budget k: $p(a|u, k)$. And the utility or reward function for sending k notification is simply $p(a|u, k)$. There are two advantages to model this conditional probability directly. First, we could leverage advanced non-linear models such as gradient boosted trees or neural networks to capture the complex interactions between organic and notification engagement, instead of being limited to linear models as in previous methods [7]. Second, we do not need to assume each notification send is independent [7, 8], since the nonlinear models could easily capture the diminishing return of increasing volume, specifically, the incremental value or utility of sending the $(k + 1)$-th notification to user u becomes $p(a|u, k + 1) - p(a|u, k)$. With this setup, the problem of notification volume optimization can be formally defined as:

$$\begin{aligned} \underset{k_u}{\text{maximize}} \quad & \sum_u p(a|u, k_u) \\ \text{subject to} \quad & \sum_u k_u \leq K \end{aligned} \tag{2}$$

Note that this formulation is very general, and can be easily extended to optimize any business metric m (*e.g.,* revenue), as long as we can learn a reward function $m(u, k_u)$ to estimate m based on the profile of user u and notification volume k_u.

Another point worth to mention is that the specific definition of $p(a|u, k_u)$ can be changed to represent various site engagement metrics that we want to optimize, *e.g.,* daily active users (DAU), weekly active users (WAU). For DAU, $p(a|u, k_u)$ is defined as the probability that the user visits the site at least once a day, and for WAU, it would be the probability that the user comes to the site at least once a week. From our experiments with various metrics, we have found out that optimizing DAU gives a good balance between active users and less engaged users. For other target metrics like WAU, it would further boost the engagement of less active users, but could hurt the performance on active users, which is not desirable since active users contribute more on revenue.

4.2 Modeling the Long Term Effect

One very important point to consider in notification volume optimization is the possible negative consequence if the volume is too high. For example, the user could unsubscribe from email or push from the settings, label emails as spam, or even delete the mobile app to stop receiving push notifications. In such case, we lose the ability to continue sending notifications to the user and will hurt user engagement as a result. Therefore, our objective function $p(a|u, k_u)$ should consider both positive and negative effect of notifications.

Previous work [7, 8] leverage Multi-Objective Optimization to solve this problem: the total number of positive actions and negative actions over all users are treated as two separate constraints, with an upper bound on the negative actions and a lower bound on the positive ones. We find there are two issues with this approach. First, specific parameters need to be set for the upper and lower bounds, and it is unclear what is the objective function to optimize when tuning these parameters in practice. Second, the cost of negative actions could be very different for different user groups, since certain users could have much higher chance to churn entirely from the site if they unsubscribe from notifications. Therefore, having a global constraint on the total number of negative actions is not able to capture this and minimize the total cost of negative actions.

To solve the above issues, in this work we propose to model the long term cost of negative actions for each user. We emphasize long term because it is aligned with our true objective and we have also observed from real data that user activities will typically continue decreasing for a few weeks after they unsubscribe. With the long term cost estimation model, and the model to predict the likelihood of unsubscribing, we will be able to approximate the long term effect of notification volume on the activeness of each user. This way, we have a unified objective function that can capture our real goal, and balance the weights of positive and negative actions at user level.

Let $p(s|u, k_u)$ denote the probability of user performing action s this week given notification volume k_u, and $\sum_s p(s|u, k_u) = 1$. And let $p(a|u, k_u, s)$ represent the effect on user activeness given action s is performed by the user. Generally speaking, the prediction of user activity becomes the expected effect of various user actions s:

$$p(a|u, k_u) = \sum_s p(s|u, k_u) \times p(a|u, k_u, s). \tag{3}$$

Note that the above setting resembles a simple Markov Decision Process (MDP) [9] in reinforcement learning, where each user action s represents a state, $p(s|u, k_u)$ is the state transition probability, and $p(a|u, k_u, s)$ is the reward function at state s. In a typical MDP setting, we would treat each week as a time step and select volume of this week k_u such that the sum of the discounted rewards over the future weeks is maximized. However, in this work we need to simplify the setup to make the problem more tractable, since in a user facing application, it is very difficult to estimate future rewards due to unknown user behavior, with the exception of certain states (*e.g.,* unsubscribe) where there is less uncertainty in future and we can make a reasonable estimation of long term reward.

Specifically, we will only consider two possible user actions each week: unsubscribe from notifications s_{unsub} and continue to subscribe s_{sub}. We will build a model to predict the probability of unsubscribing $p(s_{unsub}|u, k_u)$. To estimate the effect of unsubscribing, one way is to assume this week the volume is 0: $p(a|u, k_u = 0)$. But in most cases this is an overestimation since the activity of users who unsubscribe will typically continue dropping for a few weeks until it becomes stable. Therefore, we will learn a model to estimate the long term (*i.e.,* a few weeks later) activeness for users who unsubscribe: $p(a_L|u, s_{unsub})$, which is tractable since in the following weeks users will not receive notifications and therefore there is less uncertainty caused by the system. For users who do not unsubscribe, we only consider the immediate reward, which is the activity prediction this week: $p(a|u, k_u, s_{sub})$. This is reasonable: the user's future activity is highly uncertain and much less predictable since the user's activity in a future week is much more correlated with the notification volume in the same week, which is unknown, rather than the volume this week. To summarize, the reward function in Equation 3 can be further written as:

$$\begin{aligned} p(a|u, k_u) = \ & p(s_{unsub}|u, k_u) \times p(a_L|u, s_{unsub}) + \\ & (1 - p(s_{unsub}|u, k_u)) \times p(a|u, k_u, s_{sub}) \end{aligned} \tag{4}$$

5 MODELS AND ALGORITHMS

In this section, we will discuss the details on the three models introduced in the previous section that form our objective function: namely, the activity prediction model $p(a|u, k_u, s_{sub})$, unsubscribe prediction model $p(s_{unsub}|u, k_u)$ and unsubscribe long term effect model $p(a_L|u, s_{unsub})$. We will also introduce an efficient algorithm to compute the optimal budget allocation among users that maximizes the objective function given a constraint on total notification volume.

5.1 Models

For all the three models mentioned above, we treat them as binary classification problems with logistic loss function.

In our work, it is important to use nonlinear models such as neutral networks or gradient boosted decision trees (GBDT). Specifically, in our current system we use XGBoost [5] to train GBDT models, and we verified the prediction accuracy is indeed much higher than logistic regression. Despite that nonlinear models are generally more powerful than linear models on large scale training data, they are particularly important in our system: for activity prediction, visitations from the organic and multiple notification channels (*e.g.,* email, mobile push) are not independent and have complex interactions; also we aim to directly model the nonlinear effect of multiple notifications in a week instead of assuming each notification send is independent.

To collect unbiased training data, we randomize the notification volume for a group of users and collect their responses. Given user u and the weekly budget k_u, for activity prediction model $p(a|u, k_u, s_{sub})$, we aim to optimize daily active users, which means for each day in the week, if the user is active on the site, we generate a positive example $(+1, u, k_u)$, otherwise a negative one $(-1, u, k_u)$. Note that one additional advantage in this model is that we could capture all kinds of indirect effect of notifications on site activity that could not be directly tracked, *e.g.,* the user may see the notification, get reminded, and later directly log on to the site without clicking through the notification.

For unsubscribe prediction model $p(s_{unsub}|u, k_u)$, if the user unsubscribes in the week, we create a positive example $(+1, u, k_u)$, otherwise a negative one. Note that here it is important k_u in the training examples should be the allocated weekly budget for user u instead of the actual number of notifications sent in the week. This is to avoid survivorship bias: if the user unsubscribes, we could not send them more notifications, so if we use the actual number of notifications we will find out in training data that users with fewer number of notifications have higher ratio of unsubscribing, which is contradictory.

For the unsubscribe long term effect model $p(a_L|u, s_{unsub})$, we look at users who unsubscribe from notifications in a week, collect the number of days they are active in the 4th week after, and generate training examples in the same way as the activity prediction model. We find the 4th week is a reasonable choice since it is generally when activity becomes more stable and not too long to slow down our development iteration speed.

Also, it is worth to mention that our framework handles both emails and push notifications, and our user base can be divided into three main groups based on available notification channels: email only, push only, email & push users. Obviously, user behaviors in these three groups are significantly different, so in practice we train separate models for each user group.

5.2 Features

In general features used in all the three models are almost the same. They can be summarized in the following categories:

- user profile: country, gender, age, signup age, *etc.*
- user organic activity history: number of days they are active in various time windows, number of pins and boards created, *etc.*
- user email activity history: email open rate / click through rate in various time windows, *etc.*
- user push notification activity history: push notification open rate, user device platform (since user behaviors on different platforms are quite different), *etc.*

For the activity prediction model and unsubscribe prediction model, the weekly budget k_u is also a feature in the model. And we find it is also helpful to create explicit interaction features based on k_u, *e.g.,* expected number of clicks (*i.e.,* user historical CTR times k_u).

Obviously the above mentioned are only base features, and the GBDT model would automatically learn interactions among these features.

5.3 Optimization Algorithm

Now we have all the models needed to score the objective function, we need to solve the constrained optimization problem defined in Equation 2, and compute the weekly notification budget for each user. The optimization algorithm needs to be scalable to handle hundreds of millions of users.

Algorithm 1 Budget Allocation Algorithm

```
1:  function ALLOCATE($u, \theta$)
2:      for $i \leftarrow k_{min}, k_{max}$ do
3:          $p[i] \leftarrow p(a|u, k_u = i)$
4:      end for
5:      $i_{max} \leftarrow \arg\max_i p[i]$
6:      for $i \leftarrow k_{min}, i_{max}$ do
7:          if $p[i_{max}] - p[i] \le \theta \times (i_{max} - i)$ then
8:              return $i$
9:          end if
10:     end for
11: end function
```

We have explained that the incremental utility of sending the $(k+1)$-th notification to user u is $p(a|u, k+1) - p(a|u, k)$. Intuitively, to achieve the maximum utility with limited number of notifications, we need to remove the notifications with lowest incremental utility. Let us assume we have a global minimum threshold θ on the incremental utility of each notification, we could have a simple algorithm in Algorithm 1 to allocate weekly notification budget. For each user, we first compute the optimal budget i_{max} in the allowed range $[k_{min}, k_{max}]$, then we gradually increase budget from k_{min} to i_{max} until the average incremental value of the remaining notifications is below the threshold. This algorithm can be easily implemented using Map-Reduce to scale.

The remaining question is how we find the threshold θ such that the total number of notifications is below K. This is very simple as well (Algorithm 2), since we just need to search in a range of parameters, for each threshold run Algorithm 1 on users (or only a sample of users) to compute the total number of notifications, and choose the minimum threshold with the total number below K. Obviously, this algorithm can be easily parallelized as well.

These two simple algorithms allow us to efficiently compute weekly notification budget for hundreds of millions of users. Also computationally our algorithms are much less expensive than a quadratic programming solver needed in other systems [7, 8]. In practice, since our users grow every day, we set constraints on the

Algorithm 2 Threshold Searching Algorithm

```
1:  function SEARCH($U, K$)
2:      for $\theta \leftarrow \theta_{min}, \theta_{max}$ do
3:          for all $u \in U$ do
4:              $k_u \leftarrow$ ALLOCATE($u, \theta$)
5:          end for
6:          if $\sum_u k_u \leq K$ then
7:              return $\theta$
8:          end if
9:      end for
10: end function
```

average number of notifications instead of the total. Also as mentioned before, different user groups based on available notification channels behave very differently, so we have different constraints and run the optimization algorithms separately for each user segment.

6 EXPERIMENTS

In this section we discuss the experimental results when we launched our new volume optimization system and compared with our previous system. As explained in Section 3.4, the previous volume optimization method is also based on machine learning models, and generally aims to send more notifications to users with higher predicted notification CTR by controlling the cadence of each notification type.

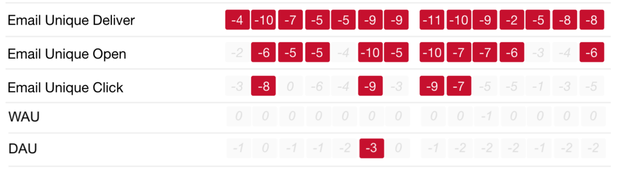
User Group	Volume%	Notification CTR%	DAU
Email Only	-24%	+31%	+0%
Push Only	-6%	+11%	+1%
Email & Push	-7% (email) -4% (push)	+10% (email) +21% (push)	+3%

Table 1: Overview of A/B test results

We conducted separate A/B experiments on three different user groups, and Table 1 is an overview of comparison between our system with the final shipped settings and the previous system. Overall, we have significantly reduced notification volume, increased CTR and also achieved significant gains on site engagement metrics (*i.e.*, DAU).

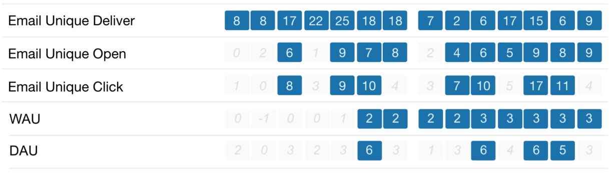
The reason we separate users based on their available notification channels is that push notifications are generally more effective than emails to bring users back to the site, so if we treat all users in the same group, the system would significantly shift notification volume from email users to push users. However, we would like to keep the email and push volume relatively balanced and also have more direct control. But for the same reason, we applied different settings for different user segments: for email only users we decided to cut the volume more aggressively since the incremental values are lower, and for push notification users we reduced less volume and gained more daily active users.

Note that although we reduced notification volume over all users, our objective function indicates that we would shift volume from more active users to less active users based on the incremental utility



Email Unique Deliver	-4	-10	-7	-5	-5	-9	-9	-11	-10	-9	-2	-5	-8	-8
Email Unique Open	-2	-6	-5	-5	-4	-10	-5	-10	-7	-7	-6	-3	-4	-6
Email Unique Click	-3	-8	0	-6	-4	-9	-3	-9	-7	-5	-5	-1	-3	-5
WAU	0	0	0	0	0	0	0	0	0	-1	0	0	0	0
DAU	-1	0	-1	-1	-2	-3	0	-1	-2	-2	-2	-1	-2	-2

Figure 5: A/B test results on core users



Email Unique Deliver	8	8	17	22	25	18	18	7	2	6	17	15	6	9
Email Unique Open	0	2	6	1	9	7	8	2	4	6	5	9	8	9
Email Unique Click	1	0	8	3	9	10	4	3	7	10	5	17	11	4
WAU	0	-1	0	0	1	2	2	2	2	3	3	3	3	3
DAU	2	0	3	2	3	6	3	1	3	6	4	6	5	3

Figure 6: A/B test results on marginal users

of notifications. This is indeed happening: Figure 5 shows the screen shot of our internal A/B testing dashboard for the comparison on core users (defined as users who saved a pin on more than 4 days in the last 28 days) in the email only segment. Each column shows the metrics on the k-th day since users are triggered into the experiment. We can see email volume and clicks are significantly reduced for core users, but there is no significant change on DAU and WAU. Figure 6 shows the results on marginal users (defined as users who were active 1-3 days in the past 28 days) in email only segment. Email volume is higher, and correspondingly email clicks, DAU and WAU metrics all increase as a result, which justifies that our system is able to shift notification volume to where the incremental utility on DAU is higher.

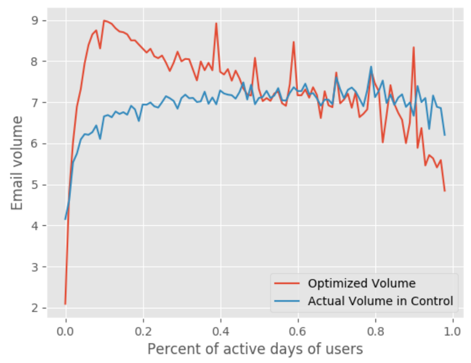
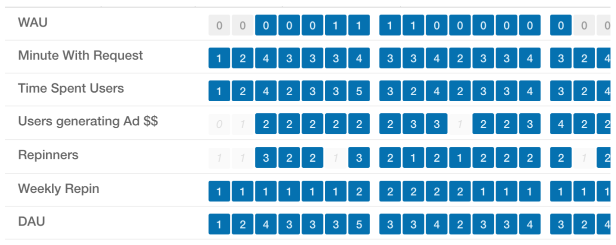


Figure 7: Volume distribution v.s. user activity level

To further illustrate this, we optimized the budget allocation with the same overall volume as in the previous production system (for email only segment). Figure 7 plots the optimized volume and actual volume in control group for different user activity level, and we can clearly see the optimized volume is lower for users who are either very active or extremely dormant, and generally higher for users who are less active. The optimized budget distribution is in similar shape to the curve of simplified utility score in Figure 4, but for very active users, the optimized budget is not approaching 0 as in Figure 4, which justifies that organic and notification channels are not independent.



WAU	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
Minute With Request	1	2	4	3	3	3	4	3	3	4	2	3	3	4	3	2	4
Time Spent Users	1	2	4	2	3	3	5	3	2	4	2	3	3	4	3	2	4
Users generating Ad $$	0	1	2	2	2	2	2	2	3	3	1	2	2	3	4	2	2
Repinners	1	1	3	2	2	1	3	2	1	2	1	2	2	2	2	1	2
Weekly Repin	1	1	1	1	1	1	2	2	2	2	2	1	1	1	1	1	1
DAU	1	2	4	3	3	3	5	3	3	4	2	3	3	4	3	2	4

Figure 8: A/B test results on email & push users

It is worth to mention that one additional advantage of our system compared with the previous system is that for email & push users, our advanced model is able to better capture the combined effect of emails and push notifications. Therefore, we saw more significant gain on push notification engagement for email & push segment (Table 1). Figure 8 also shows the gains on a few other top-line business metrics in the A/B experiment on email & push segment.

7 CONCLUSIONS

In this paper, we introduced the notification volume control and optimization system at Pinterest. We proposed a novel machine learning approach to compute weekly notification volume for each user such that long term user engagement is optimized, and a global volume constraint is met. Compared with previous methods, our framework is able to leverage more advanced nonlinear models to improve prediction accuracy without the need for certain simplified assumptions, and the optimization algorithm is very efficient and scalable to handle hundreds of millions of users. We also discussed a few practical lessons we learned and the reasoning behind a few design choices we have made. The new system has been deployed to production at Pinterest in mid 2017, and A/B experiments show that we significantly reduced notification volume and improved CTR of notifications and site engagement metrics compared with the previous system.

8 ACKNOWLEDGMENTS

We would like to thank Utku Irmak for developing the previous notification volume control and ranking system at Pinterest, and providing valuable suggestions to our work; and Seth Weisfeld for valuable insights from product and user experience perspectives.

REFERENCES

[1] Douglas Aberdeen, Ondrej Pacovsky, and Andrew Slater. 2010. The learning behind gmail priority inbox. In *in NIPS 2010 Workshop on Learning on Cores, Clusters and Clouds.*

[2] Deepak Agarwal, Bee-Chung Chen, Pradheep Elango, and Xuanhui Wang. 2012. Personalized click shaping through lagrangian duality for online recommendation. In *The 35th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '12, Portland, OR, USA, August 12-16, 2012.* 485–494.

[3] Steffen Bickel and Tobias Scheffer. 2006. Dirichlet-Enhanced Spam Filtering based on Biased Samples. In *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006.* 161–168.

[4] Dotan Di Castro, Zohar Shay Karnin, Liane Lewin-Eytan, and Yoelle Maarek. 2016. You've got Mail, and Here is What you Could do With It!: Analyzing and Predicting Actions on Email Messages. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA, February 22-25, 2016.* 307–316.

[5] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016.* 785–794.

[6] Chantat Eksombatchai, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, and Jure Leskovec. 2017. Pixie: A System for Recommending 3+ Billion Items to 200+ Million Users in Real-Time. *CoRR* abs/1711.07601 (2017). http://arxiv.org/abs/1711.07601

[7] Rupesh Gupta, Guanfeng Liang, and Rómer Rosales. 2017. Optimizing Email Volume For Sitewide Engagement. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM 2017, Singapore, November 06 - 10, 2017.* 1947–1955.

[8] Rupesh Gupta, Guanfeng Liang, Hsiao-Ping Tseng, Ravi Kiran Holur Vijay, Xiaoyu Chen, and Rómer Rosales. 2016. Email Volume Optimization at LinkedIn. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016.* 97–106.

[9] Ronald Howard. 1960. *Dynamic Programming and Markov Processes.* The M.I.T. Press.

[10] David C. Liu, Stephanie Rogers, Raymond Shiau, Dmitry Kislyuk, Kevin C. Ma, Zhigang Zhong, Jenny Liu, and Yushi Jing. 2017. Related Pins at Pinterest: The Evolution of a Real-World Recommender System. In *Proceedings of the 26th International Conference on World Wide Web Companion, Perth, Australia, April 3-7, 2017.* 583–592.

[11] Bradley Taylor. 2006. Sender Reputation in a Large Webmail Service. In *CEAS 2006 - The Third Conference on Email and Anti-Spam, July 27-28, 2006, Mountain View, California, USA.*

[12] Andrew Zhai, Dmitry Kislyuk, Yushi Jing, Michael Feng, Eric Tzeng, Jeff Donahue, Yue Li Du, and Trevor Darrell. 2017. Visual Discovery at Pinterest. In *Proceedings of the 26th International Conference on World Wide Web Companion, Perth, Australia, April 3-7, 2017.* 515–524.