An Empirical Study on Recommendation with Multiple Types of Feedback

Liang Tang
LinkedIn Corporation
Mountain View, CA, USA
ltang@linkedin.com

Bo Long
Particle Media Inc
Santa Clara, CA, USA
bo.long@particle-inc.com

Bee-Chung Chen
LinkedIn Corporation
Mountain View, CA, USA
bchen@linkedin.com

Deepak Agarwal
LinkedIn Corporation
Mountain View, CA, USA
dagarwal@linkedin.com

ABSTRACT

User feedback like clicks and ratings on recommended items provides important information for recommender systems to predict users' interests in unseen items. Most systems rely on models trained using a single type of feedback, e.g., ratings for movie recommendation and clicks for online news recommendation. However, in addition to the primary feedback, many systems also allow users to provide other types of feedback, e.g., liking or sharing an article, or hiding all articles from a source. These additional feedback potentially provides extra information for the recommendation models. To optimize user experience and business objectives, it is important for a recommender system to use both the primary feedback and additional feedback. This paper presents an empirical study on various training methods for incorporating multiple user feedback types based on LinkedIn recommendation products. We study three important problems that we face at LinkedIn: (1) Whether to send an email based on clicks and complaints, (2) how to rank updates in LinkedIn feeds based on clicks and hides and (3) how jointly optimize for viral actions and clicks in LinkedIn feeds. Extensive offline experiments on historical data show the effectiveness of these methods in different situations. Online A/B testing results further demonstrate the impact of these methods on LinkedIn production systems.

Keywords

Recommender System; Personalized Recommendation; Multi-objective Optimization

1. INTRODUCTION

Recommender systems have been applied to a wide range of applications, such as recommending news articles, movies, books, and research papers. Recommender systems seek to predict the "preference" that a user would give to an item. A typical way for recommender systems to provide recommendations is to build a model based on users' historical feedback (previously clicked, purchased or selected and/or numerical ratings given to those items); then use that model to predict items (or ratings for items) that the users may have an interest in the future.

As modern recommender systems become more complicated and touch more aspects of user experience, there has been a rapid increase in demand for a variety of user feedback to be incorporated into a recommendation model. A few examples are in the follow.

- *News feed*: The news feed of a social network site provides a user with updates from the neighbors of the user in the network. For example, a user's LinkedIn news feed contains articles and images shared, liked or commented on by other users connected to the user. To optimize for user engagement, a recommender system can be built to rank updates based on the predicted click through rate (CTR) for each (user, update) pair. While clicks provide primary feedback for the recommender system, a user can also hide updates from another user indicating strong dissatisfaction, providing additional types of feedback. When the system tries to maximize CTR, it is also important to minimize the hide rate.

- *Email campaign*: Many companies send promotion emails to users to engage with them. A recommender system can be built to decide whether to send a promotion email to a user based on the predicted CTR for each (user, email) pair in order to maximize users' responses to the emails. At the same time, it is important to minimize the chance that a user would unsubscribe all emails from the company or tag the email as a spam, indicating strong dissatisfaction. We call both unsubscriptions and spam tags *complaints*. In this case, clicks provide primary feedback, while complaints provide additional feedback.

- *Online shopping*: Whether a type of feedback is primary or secondary depends on the application. For online shopping, maximizing revenue from purchases is the primary objective. Thus, users' actual purchases provide primary feedback, while clicks on items may provide additional feedback.

Incorporating multiple types of user feedback in a recommendation model has become an important challenge for more and more recommender systems, though the related research is still relatively new in the field of recommendation. An intuitive solution to the problem is simply incorporating different feedback types into labels in training data, e.g., for click and hide feedback, we can label

KDD '16, August 13-17, 2016, San Francisco, CA, USA
© 2016 ACM. ISBN 978-1-4503-4232-2/16/08. . . $15.00
DOI: http://dx.doi.org/10.1145/2939672.2939690

data instances such that click is 1, non-click is 0, hide is -1. However, this will change the original binary classification model to a multinomial model and a multi-class prediction problem. It is not favorable for the recommendation and ranking purpose, because it generates multiple ranking scores and multiple ranking recommendations for each item. Keeping the same type of models, like binary classification model, and handling an additional feedback inside the model itself is a more preferable and scalable solution for most recommender systems. Furthermore, in a lot of situations, some additional feedback may not be able to put into labels. For example, downstream utilities such as page views are not easily put into binary labels as clicks.

In this paper, we only focus on generalized linear model based recommendation algorithms, which predict the user's interest and rank items based predicted interests. The performance of recommendation heavily depends on the accuracy of prediction. At LinkedIn, this type of algorithms is widely used in many products since content information, such as user profile and job information, are available for models. Collaborative filtering based approaches, such as matrix factorization, are mainly used in feature engineering to generate latent features for the generalized linear models.

The first method we consider is model combination, i.e., training one model for each feedback data then combining all models into a single model. This approach is feasible if the outputs of all models are the ranking scores or recommendation scores for different feedback types. For example, two logistic regression models for click data and hide data can be combined. We can either combine the predicted responses of the models or combine the model coefficients if the two models are in the same feature space and formulations are also the same. Additional weight parameters control the tradeoff of the importance of different types of feedback.

The second method is a sequential training based on Bayesian inference. In the previously mentioned example of the click feedback and hide feedback, we first train a model only using the click data and then use the model as the prior to train the final model using the hide feedback data. The final model fits the hide data but is also regularized by the click model. In general, we sequentially train the model for each feedback type where the prior is utilized to incorporate the previous model.

The third method is joint training, where we put multiple feedback types into a single training problem. The joint training still has one primary feedback to optimize, but we incorporate other secondary feedback as constraints to regularize the optimization problem. This constrained optimization framework can be applied to various recommendation applications. It is capable of incorporating some additional feedback types that are very different from the primary feedback type without changing the original recommendation model; i.e., there is no need to revise the online recommendation system to support multiple models and scorings. Meanwhile, in many applications, a certain correlation exists in different types of user feedback. For example, clicks and hides are often negatively correlated; the likes and clicks are positively correlated.

In this paper, we present an empirical evaluation on the three methods based on LinkedIn recommendation products. We study three important problems that we face at LinkedIn: (1) Whether to send an email based on clicks and complaints, (2) how to rank feed updates in LinkedIn feeds based on clicks and hides and (3) how jointly optimize for the viral actions and clicks in LinkedIn feeds. Extensive offline experiments on historical data show the effectiveness of these methods in different situations. Online A/B testing results further demonstrate the impact to LinkedIn production systems with real traffic. Meanwhile, *model combine* has been adopted by the current LinkedIn feed modeling in both desktop and mobile platforms to promote viral actions and improve the overall liquidity of the users' social network.

The rest of this paper is organized as follows. In Section 2, we formally introduce the problem of recommendation with multiple types of feedback. Detailed description for the three methods that we evaluated is presented in Section 3. Section 4 briefly discusses the implementation of the training algorithm. Extensive empirical evaluation results are reported in Section 5. Section 6 presents a brief summary of prior work relevant to the multi-feedback and multi-criteria recommendation problems, and transfer learning. Finally, Section 7 concludes the paper.

2. PROBLEM FORMULATION

Recommender systems build a model from users' historical impressions and feedback, such as clicks, hides, likes, shares, comments, and purchases etc. In LinkedIn, the recommendation models usually consider multiple user feedback. For instance, LinkedIn feeds models aim to maximize the click per impression (CTR), but also minimize the hide per impression since hide is a strong negative feedback. Meanwhile, like, comment and share are more important than click because they can significantly increase the virality of the contents and the liquidity of the social network. Hence, those actions are important feedback to maximize in recommendation as well. Therefore, our recommendation model has multiple objectives.

We denote our data as $\mathcal{P} = \{(\mathbf{x}_i, y_i^{(1)}, y_i^{(2)}, ..., y_i^{(m)})\}_{i=1}^N$, where where $\mathbf{x}_i$ is the feature vector of the i-th impression, which includes the user features and item features, $y_i^{(1)}$, ... ,$y_i^{(m)}$ are the m different types of feedback received in this impression. All the feedback are represented as binary variables: 1 or 0. For instance, $y_i^{(1)} = 1$ indicates there is a click action and 0 indicates this impression does not receive any click. The recommendation model calculates the ranking score for each impression and selects (or ranks) the items based on the ranking scores. Let $f(\mathbf{x}_i, \beta)$ be the ranking score of a given feature vector $\mathbf{x}_i$, where f is the model function and β is the coefficient vector of the model. For instance, we use the logistic regression to predict the CTR of articles. In this model, the ranking score is the predicted CTR, $f(\mathbf{x}_i, \beta) = 1/(1 + \exp(-\mathbf{x}_i^T \beta))$. Selecting the articles with high predicted CTRs can maximize the received click feedback from the users in future.

The problem that we study in this paper is to build a recommendation model that maximizes (or minimizes) m types of feedback received from users respectively, where the m types of feedback are pre-defined. For instance, for job recommendation at LinkedIn, two typical types of user feedback are click and dismiss. Click is a positive feedback that indicates the user is interested in the recommended job. However, dismiss is a strong negative feedback that indicates the user is upset by seeing this job recommendation. The desired model should be able to maximize the number of received clicks and minimize the number of received dismisses simultaneously. It is worth to mention that we still need a single model to make the recommendation decision for each job posting, which is recommend or not recommend.

3. METHODS

When the number of feedback types $m > 1$, the studied problem is a multi-objective optimization or multi-task learning problem. We explored three potential methods for solving this problem based on LinkedIn products and data. The first simple method is called *model combine*. Basically, this method trains an individual model for each objective. Since there are m feedback types, it has m individual models, $f_1, ..., f_m$. The final model is the combination of

$f_1, ..., f_m$. A simple combination is the weighted linear combination, such that the final model $f(\mathbf{x}_i, \beta) = \sum_{j=1}^{m} w_j \cdot f_j(\mathbf{x}_i, \beta_j)$, where w_j is the weight parameter to control the tradeoff between different objectives and β_j is the model coefficient for the model j, where $j = 1, ..., m$. This method can train the models in parallel and compose the final model in the end.

Besides training the individual models in parallel, we also explored a sequential training method. Let L be the loss function for the training algorithm for each feedback. We sequential solve the model coefficients $\beta_1, ..., \beta_m$ by the following series of optimization problems,

$$\min_{\beta_1} \sum_{i=1}^{N} L(\mathbf{x}_i, y_i^{(1)}; \beta_1),$$
$$\min_{\beta_j} \sum_{i=1}^{N} L(\mathbf{x}_i, y_i^{(j)}; \beta_j) + w_j ||\beta_j - \beta_{j-1}||^2,$$
$$j = 2, ..., m.$$

The final model is the last model with coefficient β_m. The idea of this method is to utilize the previous model as the prior to regularize the next model, so that the training result from the previous model can be transmitted to the next model. The final recommendation model is the last model that should contain the information of all the m models. Thus, we call this method as *prior combine*. Some literatures also mention this method is a warm-start training, in which they consider the old model as the prior or starting point, and use the new training data to train the new model. It is worth to mention that the method has two assumptions. First, the prior model's coefficient is a multivariate normal distribution and second the variance of each dimension is identical and determined by the inverse of w_j. However, the assumption may not exist in many high dimension data. In practice, it is very common that some of the features are sparse and some of the features are dense in real world high dimensional data. Thus, the associate coefficients have very distinct variances.

The third method is the joint training, where all types of feedback are utilized in a single joint optimization problem for the final model. The general form is,

$$\min_{\beta} \sum_{i=1}^{N} L(\mathbf{x}_i, y_i^{(1)}, ..., y_i^{(m)}; \beta),$$

where L is the joint loss function. There are lots of different approaches to compose the joint loss L. For instance, L is a linear combination of a logistic loss and a constraint loss (e.g., the hinge loss). In this instance,

$$\begin{aligned} L(\mathbf{x}_i, y_i^{(1)}, y_i^{(2)}) = \quad & w_1 \cdot \log(1 + \exp(-y_i^{(1)} \cdot f(\mathbf{x}_i, \beta))) \\ & + w_2 \cdot \max(0, y_i^{(2)}(c + f(\mathbf{x}_i, \beta))), \end{aligned}$$

where w_1, w_2 are the weight parameters, c is a given positive parameter, and $y_i^{(2)}$ is a strong negative feedback from users (e.g., hide and dismiss). The hinge loss, $\max(0, y_i^{(2)}(c + f(\mathbf{x}_i, \beta)))$, forces the ranking score $f(\mathbf{x}_i, \beta) \leq -c$ if $y_i^{(2)} = 1$. Thus, it is a constrained optimization. It aims to minimize the loss from the primary feedback $y_i^{(1)}$ subject to the constraints $f(\mathbf{x}_i, \beta) \leq -c$ if $y_i^{(2)} = 1$. The weight w_2 determines the hardness of the constraints. If w_2 is an arbitrary large number, the constraints are hard constraints. Similarly, if $y_i^{(2)}$ is a strong positive feedback from users (e.g., like or apply a job), an example of L is

$$\begin{aligned} L(\mathbf{x}_i, y_i^{(1)}, y_i^{(2)}) = \quad & w_1 \cdot \log(1 + \exp(-y_i^{(1)} \cdot f(\mathbf{x}_i, \beta))) \\ & + w_2 \cdot \max(0, y_i^{(2)}(c - f(\mathbf{x}_i, \beta))), \end{aligned}$$

where the hinge loss, $\max(0, y_i^{(2)}(c - f(\mathbf{x}_i, \beta)))$, forces the ranking score $f(\mathbf{x}_i, \beta) \geq c$ if $y_i^{(2)} = 1$. It is common that a recommendation model has one primary type of feedback (e.g., click) with other secondary types of feedback (e.g, hide and dismiss). The secondary types of feedback are either strong positive or strong negative, so we utilize them as constraints to regularize the primary feedback optimization problem. This method is called *constrained regression* in this paper. In many applications, different types of user feedback may have some correlation. For instance, in the LinkedIn feeds, the click and like are positive correlated, while the click and hide are negative correlated. In these situations, the constraints from the secondary types of feedback can help the learning algorithm better regularize the primary feedback learning. At the same time, the primary feedback also helps the learning algorithm find more reliable solutions subject to the constraints. This correlation of feedback is quite useful in some practical scenarios because the strong negative or positive feedback is often very rare in online systems. For instance, the total number of received clicks can be more 100 times larger than the total number of received hides in a same time period. Such rare feedback can hardly build a good recommendation model. But if we add a large amount of correlated feedback data to train jointly, the model performance can be improved. In machine learning, this technique is defined as *Transfer Learning* [8, 32, 10, 2, 3, 4, 27], in which each feedback refers to a domain. In the experimental section, we will discuss several cases where this transfer learning works and several other cases where it does not work based on LinkedIn data. Table 1 shows several loss functions that are widely used in our recommendation models, where y_i is the label of a feedback and $\mathbf{x}_i$ is the corresponding feature vector of the impression. The last loss is a pairwise loss, where $\mathbf{x}_i'$ is the feature vector of another impression.

Table 1: Loss Function

Loss Function	**Loss Function Name**
$\log(1 + \exp(-y_i f(\mathbf{x}_i, \beta)))$	Logistic Loss
$\lvert y_i - f(\mathbf{x}_i, \beta)\rvert^2$	Square Loss
$1 - I(sign(y_i \cdot f(\mathbf{x}_i, \beta)))$	0-1 Loss
$\max(0, y_i(c + f(\mathbf{x}_i, \beta)))$	Hinge Loss
$\max(0, y_i(c + f(\mathbf{x}_i, \beta)))^2$	Squared Hinge Loss
$\max(0, y_i(f(\mathbf{x}_i, \beta) - f(\mathbf{x}_{i'}, \beta)))$	Pairwise Hinge Loss

The three methods have various advantages and disadvantages in different data sets. *prior combine* takes the trained results from the previous model to regularize the next model's fitting. It has an initiative explanation based on Bayesian inference. *model combine* trains different models separately, where each model only has one objective so its training can easier coverage to its own optimal solution. *constrained regression* is a joint training, which utilizes multiple types of feedback together and is potentially to obtain a better joint optimal solution. It is difficult to theoretically determine which method is the best one in different particular cases. In this paper, we only focus on the empirical evaluation based on LinkedIn data.

4. IMPLEMENTATION

The offline training data is collected from a random bucket of users, where the historical recommended items are randomly selected to the users in order to avoid the sample bias and serving bias. In the algorithm implementation, we only focus on the convex loss function of L. In order to speed up the offline model training, we apply the coordinate gradient descent algorithm [37]. This algorithm minimizes the total loss function along one dimension at a time. On each dimension, it only goes one step along the opposite gradient direction. Recent studies have shown that the efficiency of this coordinate-wise optimization method is superior to many convex optimization algorithms in practice [37, 16]. Another advantage of this algorithm is that, it does not need to employ the backtracking linear search [31] to calculate the appropriate step size. Instead, it fixed the step size to be the inverse of the Lipschitz constant [5] of ∇L. The Lipschitz constant of ∇L provides an upper bound of the step size on each dimension, which is not conservative in coordinate-wise algorithms. We found this trick can significantly improve the efficiency of the algorithms in our model training.

Let d be the dimensionality of the data space, $\beta^{(i)}$ denote the i-th element of the vector β, and l_i be the Lipschitz constant of $\frac{\partial F}{\partial \beta^{(i)}}$, $i = 1, 2, ..., d$. Algorithm 1 presents the pseudocode of the

Algorithm 1 Coordinate Gradient Descent with Lipschitz Constant

```
    Input: $\mathcal{P}$: training data, $\epsilon$: tolerance parameter.
1:  $\beta \leftarrow \mathbf{0}$
2:  $||\nabla L||_{init} \leftarrow ||\nabla L(\beta)||$
3:  while $||\nabla L(\beta)|| > \epsilon ||\nabla L||_{init}$ do
4:      for $i = 1, 2, ..., d$ do
5:          $\beta^{(i)} \leftarrow \beta^{(i)} - \frac{1}{l_i} \cdot \frac{\partial L}{\partial \beta^{(i)}}$
6:      end for
7:  end while
```

algorithm, where each dimension's Lipschitz constant l_i is precomputed before entering the algorithm, ϵ is the tolerance parameter for the stopping criteria, which is the same definition for libLinear and libSVM [1]. This algorithm is flexible since it only requires the first order information of the total loss function L and the Lipschitz constant of ∇L. In this paper, for the offline experiments, the training algorithm is implemented in a single machine. For the online A/B testing, we apply the large scale of distributed optimization algorithm ADMM to scale up the production model training [**?**].

5. EXPERIMENTS

We present the empirical evaluation to answer the following important questions for recommendation with multiple types of feedback. First, what kind of feedback is feasible to be incorporated in a practical recommendation model? Second, how to select a suitable method for a particular application? Third, how do the different approaches perform for the real applications?

5.1 Complaints in LinkedIn Emails

Email is one of the most useful channel to improve the user engagement in social network industry. LinkedIn email recommender system aims to rank LinkedIn email candidates, such as news article content, "people you may know" and job postings, and then send top ranked emails to members. The primary feedback is click/non-click from members after they receive the emails. At the same time, the system records other types of feedback, such as complaints. The complaints include the actions of unsubscribing and labeling emails as spams. While the system optimizes CTR to boost member engagement, it is also desirable to minimize the complaints from the users. As described in Section 3, in this problem the number of types of user feedback is $m = 2$, click and complaint. The objective is to maximize the CTR and minimize the complaint rate for each sent email.

[1]https://www.csie.ntu.edu.tw/ cjlin/liblinear/

We investigate the three potential methods described by Section 3 in the email recommendation experiment. For *prior combine*, we implement two algorithms, "Prior Combine(Click)" and "Prior Combine(Complaint)", where the first one uses the click data to train the prior model and the second one uses the complaint data to train the prior model. For *constrained regression*, we also implement two algorithms, "Constrained(Complaint)" and "Constrained(Click Complaint)", where the first one only uses the complained emails as the constraints and the second one uses both clicked and complaints emails to establish the constraints. We tested the squared hinge loss and logistic loss for the click and complaint feedback, but there is no significant difference for the experimental results. For each algorithm, we vary the weight parameter w_j and regularization weight λ for model training. As a result, for each approach, we obtain 60 - 70 models based on different parameter combinations.

5.1.1 Data

The training and testing data is from the historical logs of a random bucket, i.e., the system randomly sent emails to a group of members and recorded their feedback. Thus, there is no serving bias for the training and testing data. We use around 2.5K features to build the recommendation model for this study. The features mainly consists of the member features, content features and interaction features. The member feature includes the member's profile data, such as the job title, education, company and industry category. The content features describe the content of the emails that we want to notify the member. The interaction feature represents the interests of a particular type of member to a particular category of emails. Table 2 summarizes the details of the data sets. We split the training data and testing data by the member Ids, so that there is no member who both appears in the training and testing data.

Table 2: Email Relevance Modeling Data Summary

Feedback	#Members for Training	# Members for Testing
Click	350K	1.2M
Complaint	460K	1.3M

5.1.2 Performance Metrics

The evaluation metric is AUC (Area Under ROC curve) [**?**]. An intuitive explanation of the AUC is the probability of a positive instance being ranked higher than a negative instance. Since there are two types of feedback, we consider the click AUC and non-complaint AUC at the same time to evaluate the model performance. The non-complaint is the opposite of the complaint. If the user did not complaint an email, the non-complaint feedback is 1, otherwise it is 0. Minimizing the complaint rate is equivalent to maximizing the non-complaint rate. A clicked email should have a higher ranking score than a non-clicked email. A non-complained email also should have a ranking higher core than a complained email. Therefore, click AUC and non-complaint AUC are higher,then the model performance is better. If model A has a both higher click AUC and non-complaint AUC than another model B, then we are sure that model A is better than model B. In multi-objective optimization, we say model A dominates model B [33]. If model A has a higher click AUC but a lower non-complaint AUC than model B, then we do not know A is better or worse than B. In this paper, our goal is to evaluate the potential training methods, not particular models, because two models with different tradeoffs may not be comparable. Since we vary different weights of click and non-complaint feedback, the tradeoff between the click AUC and the non-compliant AUC is also varied. If an algorithm P generates more models that dominate the models generated by algorithm Q, then we can say the algorithm P is better than Q in most cases.

5.1.3 Experimental Results

Figure 1 shows the click AUC and non-complaint AUC of each training method on the test data sets. To protect the user privacy, we show the experiment results in terms of normalized AUC. The normalized click AUC is the click AUC divided by the click model's AUC, where the click model is trained only by the click data. Likewise, the normalized non-complaint AUC is the non-complaint AUC divided by the complaint model's AUC, where the complaint model is trained only by the complaint data.

As shown by Figure 1, we observe that incorporating complaint feedback does not hurt our model's CTR performance too much, but it significantly decreased the complaint rate more than 40% in some area. The *constrained regression* and *model combine* have the best tradeoff between the click AUC and non-complaint AUC. They can improve the non-complaint AUC from 0.4 to 0.7 by sacrificing the click AUC as little as 0.18. For other methods, to achieve the same non-compliant AUC, they have to sacrifice larger click AUC. It is worth to note that the *constrained regression* with only complaint constraints performs worse rapidly when the weight for complaints becomes large. This is because when the complaint weight is larger, the regression algorithm spends more much effort to fit the complaint data points. As a result, the entire data set becomes highly imbalance with more negative feedback. If we put both click and complaint data points into the constraints, constraints set are well balanced as shown by "Constrained(Click Complaint)", then it performs better than "Constrained(Complaint)". The performance of *data combine* approach is almost identical to *constrained regression*. The two algorithms based on *prior combine* perform much worse than the other algorithms. This is because this *prior combine* method only utilizes the coefficient mean of the prior model into the L2 regularizer for the next model training. As we discussed in Section 3, the assumption that the prior model coefficient is a multivariate normal distribution and each dimension has an identical variance may not hold. In the email data, some member and content features are very sparse or very dense so that the corresponding coefficients should have very different variances. Using an inappropriate prior can make the final model worse.

In practice the size of the secondary feedback varies a lot in different applications and time periods, to illustrate the the situation that the secondary feedback data is scarce, we randomly sample only 100 complained emails for training in the experiments. Figure 2 shows another AUC comparison for all the methods. As shown in the figure, *constrained regression* shows a better tradeoff curve of click AUC and non-complaint AUC than other methods. In other words, for the same AUC on non-complaints, *constrained regression* models show a higher AUC on the click feedback. For example, for the models with non-complaint AUC 0.85, the *constrained regression* models achieve around 0.99 on normalized click AUC, while the *model combine* models only achieve around 0.96. Since there are only 100 complained data points, the model for predicting complaint feedback has a high variance, which also hurts the performance of the combined model. A few outliers in the 100 data points can ruin the entire compliant model in *model combine*. On the other hand, *constrained regression* is a joint training that uses the click data and complaint data together. The large number of click data can regularize the joint model to be more robust to these outliers in complaint data. By this way, the correlation between the click feedback and complaint feedback help *constrained regression* achieve better performances than *model combine*.

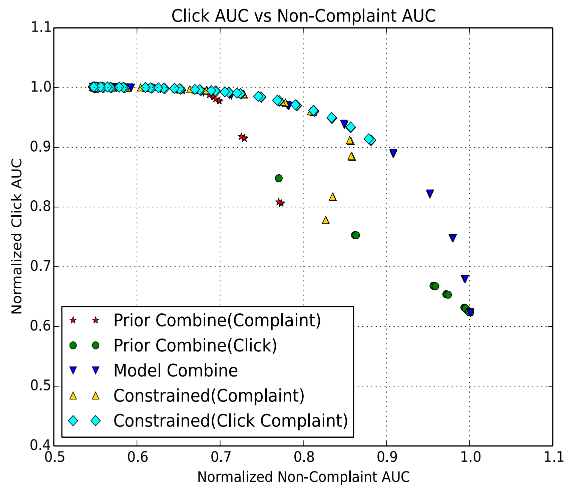


Figure 1: AUC Tradeoff on Email Data

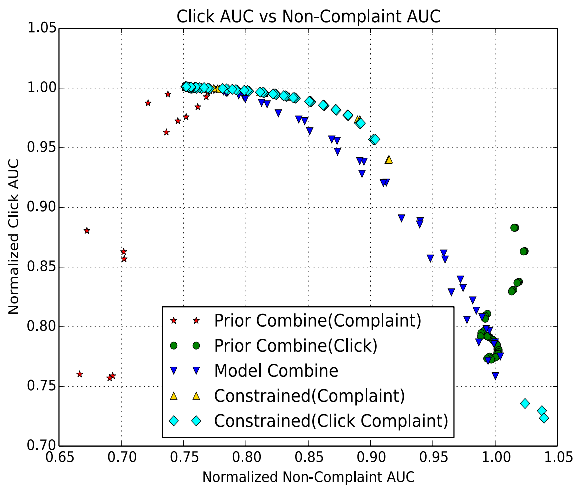
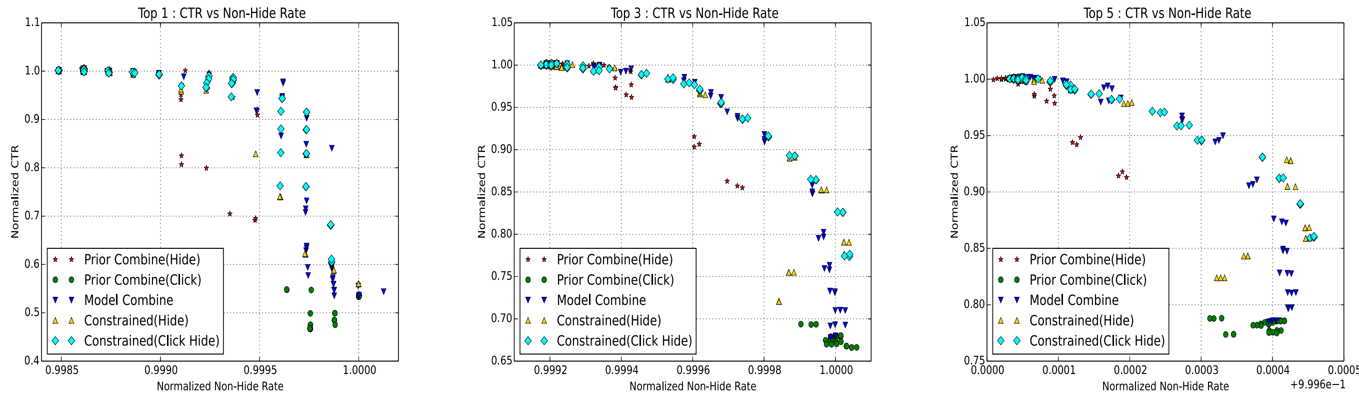


Figure 2: AUC Tradeoff on Email Data (100 Complaints in Train)

5.2 Hides in LinkedIn Feeds

LinkedIn feeds are mainly ranked by relevance models, which aim to maximize the user engagement and other business metrics. The user engagement is represented by the number of user clicks on feed items, which also include like, share, comment and other social actions. We apply the logistic regression model to predict the click probability for each item, and then rank all feed items based on the predicted click probabilities in descending order. Beside like, comment, share and other positive feedback, the users can also provide negative feedback by hiding certain feed items if they do not want to see certain feed updates again. The hide action is a strong negative feedback for user experiences. Therefore, while the ranking system optimizes CTR of feeds, it is also important to eliminate the future user's hide feedback to minimize the negative impact to user experience.



(a) Top 1 Position (b) Top 3 Positions (c) Top 5 Positions

Figure 3: CTR and Non-HideRate Tradeoff on Feed Data

We investigate the three methods described by Section 3 in this experiment. Like the LinkedIn email experiment, for *prior combine*, we implement two algorithms, "Prior Combine(Click)" and "Prior Combine(Hide)", where the first one uses the click data to train the prior model and the second one uses the hide data to train the prior model. For *constrained regression*, we also implement two algorithms, "Constrained(Hide)" and "Constrained(Click Hide)", where the first one only uses the hidden impressions as the constraints and the second one uses both clicked and hidden impressions to establish the constraints.

5.2.1 *Offline Experimental Results*

The training data and testing data are summarized in Table 3. Both data sets are sampled from the historical impressions of a random bucket, where we randomly shuffle the ranking list of feeds to a small group of users and collect the feedback in order to avoid the sample bias. The number of features is around 4.5K, which include the member profile features, item's features, and member's past behaviors etc. [**?**] lists the details of the LinkedIn feeds ranking model in production systems. Since in feed ranking, different

Table 3: Click and Hide Feed Modeling Data Summary

Feedback	#Impression for Training	#Impression for Testing
Click	810K	64.6M
Hide	410K	1.3M

feed positions have different biases to the CTR and hide rate, we select to use *precision at top* K as the evaluation metric instead of AUC, because AUC estimates the classification accuracy of items without considering the feed position. Based on the ranking order of feeds from each model, the CTR and hide rate on top K positions are calculated for each model. If a model A has a higher CTR and a lower hide rate than model B by *precision at top K*, then the model A is better than model B, and A dominates B. For each potential method, we vary different weight parameters and also the regularization weights for model training. As a result, we generate 60 - 70 models for each method based on different parameter combinations.

Figure 3 shows the CTRs and non-hide rates for top 1, 3 and 5 positions by *precision at top* K, $K = 1, 3, 5$ respectively. The non-hide feedback is the opposite of the hide feedback. To protect the user privacy, we only show the normalized CTR and non-hide rate in this paper. The normalized CTR is the CTR divided by the click model's CTR, where the click model is trained only based on the click data. Likewise, the normalized non-hide rate is the non-hide rate divided by the hide model's non-hide rate, where the hide model is trained only based on the hide data. The comparison results are similar as in the experiment for the email recommendations. *prior combine* performs much worse than other methods. *constrained regression* shows slight better tradeoffs than *model combine* on the high non-hide rate areas. It is also interesting to see that, some *constrained regression* models have higher non-hide rates than the hide model that is trained purely based on the hide feedback data. This observation reveals the effect of transfer learning, because the click action and non-hide action have a certain correlation. The large mount of click feedback data helps the joint model to build a better prediction on hide actions.

5.2.2 *Online A/B Testing Results*

To further study the performance of these methods, we conduct online A/B testing experiments in LinkedIn production systems. We launch 4 models of *constrained regression*, where each model is randomly allocated to serve a certain percentage of LinkedIn desktop users. It is worth to note that the members in the training data has no intersection with the members in the online testing. This A/B testing lasts for one week. We set the weight parameter w_2 for the constraints according to α:

$$w_2 = \begin{cases} 0 & \text{if } \alpha = 0 \\ \frac{\alpha N^-_{click}}{N^-_{hide}(1-\alpha)} & \text{if } \alpha > 0, \end{cases} \qquad (1)$$

, where N^-_{click} is the number of negative click data points and N^-_{hide} is the number of negative hide data points (or the hided items) in the training data. Meanwhile, $w_1 = 1 - w_2$ is the weight for the logistic loss of the click data. α can be explained as the ratio of the constraint loss to the entire loss from negative data, therefore, it is more intuitive than directly setting up the w_2.

The online testing experiment is deployed on two types of LinkedIn homepage. The first one is the old LinkedIn homepage, which has lower CTRs and hide rates, but the online traffic is large. The other one is the new homepage, which has a smaller online traffic but higher overall user engagements in terms of CTRs and hide rates. Tables 4 and 5 show the online results of the online experiment in the two types of LinkedIn homepage respectively. The control model is the one with $\alpha = 0$, i.e., the control model is trained only with the click data. The CTR and hide rate are calculated in

Table 4: Feed Online A/B Testing Result in Old Homepage

α	CTR Lift	Hide Rate Reduction
0	0	0
0.1	1.1%	1.8%
0.2	0.8%	4.2%
0.5	0.02%	3.6%

Table 5: Feed Relevance Online A/B Testing in New Homepage

α	CTR Lift	Hide Rate Reduction
0	0	0
0.1	2.6%	6.4%
0.2	2.0%	5.0%
0.5	5.2%	12.5%

the page view level, which are the total number of clicks or hides divided by the total number of page views. As shown in the two tables, the CTRs have a small positive lift compared against the control model. It shows that the hide feedback provides additional signals about the member's interest that benefit the click modeling. Meanwhile, the three experimental models decrease the hide rate from 1 to 12 percent. Although the effect of hide rate decreasing is not as large as offline experiments, it is clear enough to demonstrate that this method can successfully incorporate the hide feedback without losing CTR in a real online system.

5.3 Viral Action in LinkedIn Feeds

Like, share and comment are called viral action in LinkedIn feeds, because these actions can generate the new feed updates for the user's neighbors, such as "X shared an article Y", "Y liked an post of Z". Therefore, encouraging viral actions can improve the liquidity of the social network. In LinkedIn feeds, viral actions are more important than the other click actions, such as pure click, connect, follow and play. However, the volume of pure click feedback is much larger than the volume of viral actions in LinkedIn feeds. We cannot only use the viral action feedback to train the ranking model because it would lose plenty of useful signals from the clicks. It is worth to mention that the viral action is a subset of the click action. There are other types of click actions that do not generate new feed updates in the social network.

The objective of the ranking model is to maximize the CTR and viral action rate simultaneously. Compared with the hide action, the viral action is more correlated with the click action since the viral action is a subset of the click action.

Table 6 summarizes the volume of the training data and testing data in the experiment. The features of the feed ranking model are same as in the previous experiment for hide actions. The details of the model can be found in [**?**].

Table 6: Click and Viral Action Feed Data Summary

Feedback	#Impression for Training	#Impression for Testing
Click	1M	29.5M
Viral Action	1M or 2M	29.5M

In this experiment, the *constrained regression* algorithm considers the click data as the primary feedback and put viral actions as positive constraints. *prior combine* still has two algorithms that consider the click model and viral action model as the prior respectively. We vary the weight parameters to obtain various tradeoff between the CTR and viral action rate for each algorithm.

The performance metric is still the *precision at top K* for clicks and viral actions. Due to the space limitation, we only show the experimental results of $K = 1$.

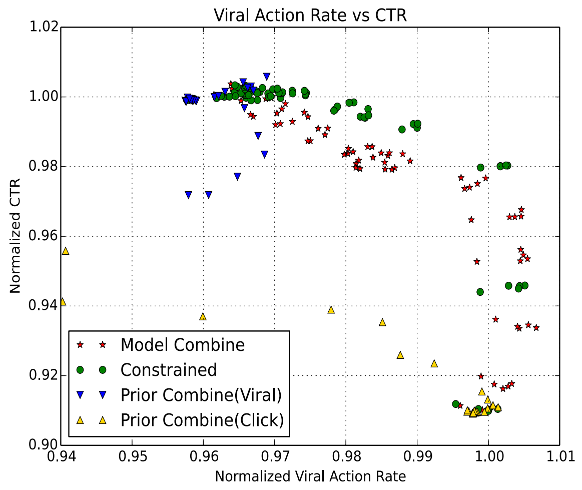


Figure 4: Top 1 Position : CTR and Viral Action Rate (1M Viral Actions in Train)

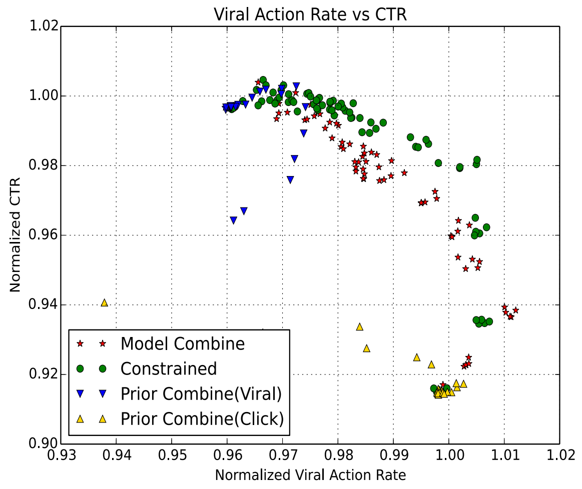


Figure 5: Top 1 Position : CTR and Viral Action Rate (2M Viral Actions in Train)

Figures 4 and 5 show the experimental results for all experimental models, where the number of viral action training data is 1M and 2M respectively in the two figures. To protect the user privacy, we only show the normalized CTR and normalized viral action rate. The normalized CTR is the CTR divided by the click model's CTR, where the click model is trained only based on the click data. Likewise, the normalized viral action rate is the viral action rate divided by the viral model's viral action rate, where the viral model is trained only based on the viral action data. It is clear to see that *constrained regression* achieved a better tradeoff than other algorithms in both figures. For example, to achieve 1.0 normalized viral action rate, *constrained regression* models can maintain 98% of the original CTR, however, *model combine* needs to lose 4% CTR.

Recall that in the previous experiment on hide actions, *constrained regression* and *model combine* have similar performances. However, in this experiment, *constrained regression* performs better than *model combine*. It is probably because the correlation between the click and viral action is larger than the correlation between the click and hide action. Therefore, *constrained regression* obtains a bigger gain from the joint training. Currently, LinkedIn feed production system adopts the *model combine* method to promote viral actions in both desktop and mobile platforms. The major reason is that, *model combine* only needs to train two individual models and vary the weights to obtain various tradeoffs, but *constrained regression* requires to retrain the joint model for each weight parameter and tradeoff. In future, we will conduct the A/B testing experiments to evaluate *constrained regression* for the click/viral action tradeoff in production systems.

In summary, for the three potential methods, *model combine* and *constrained regression* have a clear better performance than *prior combine*. If the multiple types of feedback has a certain correlation and some type of feedback has very small data, *constrained regression* performs better than *model combine*.

6. RELATED WORK

Multi-Criteria Recommendation: Many product recommender systems has different ratings for one product, such as story rating and the visual effect rating for one movie. Users also have different interests and preferences on those aspects. The goal of multi-criteria recommender systems is to maximize interests of all aspects interests at the same time. To achieve this goal, collaborative filtering based algorithms and content based algorithms have different approaches. Collaborative filtering based algorithms construct a rating vector of a user provide for an item. Each user's neighbors are computed based on their rating vectors to items, instead of a single rating variable [24, 1]. The preferences on different rating aspects can be substitute as the weights into the distance function of two rating vectors. Content based algorithms are based on the feature based learning models to predict the user interests. When the user has multiple interests, it can naturally build multiple learning models for predicting those interests [33]. For each user, the algorithms selects the Pareto optimal items to recommend [30]. Those work are similar to the model combine and response combine methods in this paper. They do not explore the benefit of the dependent responses to the user modeling. Besides the accuracy of the user interests, many research studies also consider some item's criteria, such as the item diversity [40, 25] and the novelty [23, 20, 15]. Other criteria including confidence, trust, risk, robustness, and privacy are also mentioned in the survey [35]. Those criteria are the constraints for the recommended items. Our work in this paper focuses on the criteria from the user's explicit feedback or response.

Transfer Learning: Another related field is transfer learning.Transfer learning approaches can be mainly categorized into three classes. A popular class of transfer learning methods is instance-based [6, 11, 28, 7, 19, 13, 36], which assumes that certain parts of the data in the source domain can be reused for the target domain by re-weighting. [21] proposed a heuristic method to remove "misleading" training instances from source domain so as to include "good" instances from labeled source-domain instances and unlabeled target-domain instances. [11] introduced a boosting algorithm, TrAdaBoost, which assumes that the source and target domain data use exactly the same set of features and labels, but the distributions of the data in the two domains are different. TrAdaBoost attempts to iteratively re-weight the source domain data and target domain data to reduce the effect of the "bad" source data while encourage the "good" source data to contribute more for the target domains. [6] proposed a framework to simultaneously re-weight the source domain data and train models on the re-weighted data with a kernel logistic regression classifier.

Another category of approaches can be viewed as model-based approaches [34, 26, 14, 9], which assumes that the source tasks and the target tasks share some parameters or priors of their models. An efficient algorithm MT-IVM [26], which is based on Gaussian Process (GP), was proposed to handle multi-domain learning case. MT-IVM tries to learn parameters of GP over multiple tasks by assigning the same GP prior to the tasks. Similarly, Hierarchical Bayes (HB) is used with GP for multi-task learning [34]. [14] borrowed the idea of [34] and used SVMs for multi-domain learning. The parameters of SVMs for each domain is assumed to be separable into two terms: a common term across tasks and a task specific term. [29] proposed a consensus regularization framework for transfer learning from multiple source domains to a target domain.

The third category of transfer learning approaches are feature based. [8, 32, 10, 2, 3, 4, 27], where a feature representation is learned for the target domain and used to transfer knowledge across domains. A structural correspondence learning (SCL) algorithm [8] is proposed to use unlabeled data from the target domain to extract features so as to reduce the difference between source and target domains. A simple kernel mapping function is introduced in [12], which maps the data from both domains to a high-dimensional feature space. [32] proposed to apply sparse coding, an unsupervised feature construction method, to learning higher level features across domain. On the other hand, heterogeneous transfer learning starts to attract attention very recently. We notice that [39] extends PLSA to a specific application, using social Web data to help image clustering; [38] proposes a manifold alignment based approach for heterogeneous domain adaptation; [18] formulates heterogeneous transfer learning as multi-task and multi-view learning and proposes a graph-based solution; [17] focus on single task learning with multiple outlooks, which is also related to heterogeneous transfer learning.

7. CONCLUSION

Learning multiple user feedback in a recommendation model has become an important challenge for recommender systems. Related research on multiple feedback is relatively new in the empirical study of recommender systems. This paper presents an empirical study on the methods for incorporating multiple types of user feedback into a single recommendation model. We investigate three common training methods, *prior combine*, *model combine* and *constrained regression* in this study. We conduct extensive experiments based on LinkedIn products and real historical data. In the experiments, we show how to use the methods to find the various tradeoff of the click/complaint in LinkedIn email recommendation, click/hide and click/viral action in LinkedIn feeds ranking. Based on our experimental results, *prior combine* does not perform well since its assumption for the prior model and prior distribution often does not hold in many real data. If each user feedback has enough training data, the performance of *model combine* and *constrained regression* are very close. But if some feedback data is rare, another feedback data is large, and the two types of feedback have a certain correlation, then *constrained regression* performs better than *model combine*. In this situation, *constrained regression* is actually doing a transfer learning. The correlated two types of feedback data help the model to better regularize the fitting on each other.

In addition to the offline experiments with historical data, we also conduct the online A/B testing for evaluating the *constrained regression* method for click/hide problem in LinkedIn feeds ranking based on a large amount of real users. The online A/B testing results also confirm the effectiveness of this method. The launched models can reduce 12% of the hide rate without hurting the CTR comparing against the control model.

As for the future work, we consider more advanced joint training methods to incorporate multiple types of feedback into recommendation, such as the constrained matrix factorization in collaborative filtering based on recommendation. Also, we continue to explore more advanced methodologies from transfer learning into the multiple feedback recommendation and utilize the relationship of different types of feedback to provide more accurate personalized recommendation. Meanwhile, we will continue to explore and solve other problems of the multiple types of feedback recommendation. For instance, in LinkedIn job recommendation, the user can apply a job or dismiss the job posting, where apply/not apply is the primary feedback and dismiss/not dismiss is a secondary feedback.

8. REFERENCES

[1] G. Adomavicius and Y. Kwon. New Recommendation Techniques for Multicriteria Rating Systems. *IEEE Expert / IEEE Intelligent Systems*, 22:48–55, 2007.

[2] R. Ando and T. Zhang. A high-performance semi-supervised learning method for text chunking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 1–9. Association for Computational Linguistics Morristown, NJ, USA, 2005.

[3] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems: Proceedings of the 2006 Conference*, page 41. MIT Press, 2007.

[4] A. Argyriou, C. Micchelli, M. Pontil, and Y. Ying. A spectral regularization framework for multi-task structure learning. *Advances in Neural Information Processing Systems*, 20, 2008.

[5] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[6] S. Bickel, M. Brückner, and T. Scheffer. Discriminative learning for differing training and test distributions. In *Proceedings of the 24th international conference on Machine learning*, pages 81–88. ACM New York, NY, USA, 2007.

[7] J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. Wortman. Learning bounds for domain adaptation. *Advances in Neural Information Processing Systems*, 20, 2008.

[8] J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*, 2006.

[9] E. Bonilla, K. Chai, and C. Williams. Multi-task gaussian process prediction. *Advances in Neural Information Processing Systems*, 20:153–160.

[10] W. Dai, G. Xue, Q. Yang, and Y. Yu. Co-clustering based classification for out-of-domain documents. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 210–219. ACM New York, NY, USA, 2007.

[11] W. Dai, Q. Yang, G. Xue, and Y. Yu. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200. ACM New York, NY, USA, 2007.

[12] H. Daume. Frustratingly easy domain adaptation. In *Annual meeting-association for computational linguistics*, volume 45, page 256, 2007.

[13] H. Daume III and D. Marcu. Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research*, 26:101–126, 2006.

[14] T. Evgeniou and M. Pontil. Regularized multi-task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM New York, NY, USA, 2004.

[15] F. Fouss and M. Saerens. Evaluating performance of recommender systems: An experimental comparison. In *Web Intelligence and Intelligent Agent Technology, 2008. WI-IAT'08. IEEE/WIC/ACM International Conference on*, volume 1, pages 735–738. IEEE, 2008.

[16] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

[17] M. Harel and S. Mannor. Learning from multiple outlooks. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 401–408, New York, NY, USA, June 2011. ACM.

[18] J. He and R. Lawrence. A graph-based framework for multi-task multi-view learning. In L. Getoor and T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 25–32, New York, NY, USA, June 2011. ACM.

[19] J. Huang, A. Smola, A. Gretton, K. Borgwardt, and B. Scholkopf. Correcting sample selection bias by unlabeled data. *Advances in neural information processing systems*, 19:601, 2007.

[20] N. Hurley and M. Zhang. Novelty and diversity in top-n recommendation–analysis and evaluation. *ACM Transactions on Internet Technology (TOIT)*, 10(4):14, 2011.

[21] J. Jiang and C. Zhai. Instance weighting for domain adaptation in NLP. In *Annual meeting-assosciation for computational linguistics*, volume 45, page 264, 2007.

[22] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pages 133–142, 2002.

[23] N. Kawamae. Serendipitous recommendations via innovators. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 218–225. ACM, 2010.

[24] K. Lakiotaki, N. F. Matsatsinis, and A. Tsoukiàs. Multicriteria user modeling in recommender systems. *IEEE Intelligent Systems*, 26(2):64–76, 2011.

[25] N. Lathia, S. Hailes, L. Capra, and X. Amatriain. Temporal diversity in recommender systems. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 210–217. ACM, 2010.

[26] N. Lawrence and J. Platt. Learning to learn with the informative vector machine. In *Proceedings of the twenty-first international conference on Machine learning*. ACM New York, NY, USA, 2004.

[27] S. Lee, V. Chatalbashev, D. Vickrey, and D. Koller. Learning a meta-level prior for feature relevance from multiple related tasks. In *Proceedings of the 24th international conference on Machine learning*, pages 489–496. ACM New York, NY, USA, 2007.

[28] X. Liao, Y. Xue, and L. Carin. Logistic regression with an auxiliary data source. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, volume 22, page 505, 2005.

[29] P. Luo, F. Zhuang, H. Xiong, Y. Xiong, and Q. He. Transfer learning from multiple source domains via consensus regularization. In *CIKM '08: Proceeding of the 17th ACM conference on Information and knowledge management*, pages 103–112, New York, NY, USA, 2008. ACM.

[30] K. Miettinen. *Nonlinear Multiobjective Optimization*. International Series in Operations Research & Management Science. Springer US, 1999.

[31] J. Nocedal and S. Wright. *Numerical Optimization*. Springer series in operations research and financial engineering. Springer, 1999.

[32] R. Raina, A. Battle, H. Lee, B. Packer, and A. Ng. Self-taught learning: Transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pages 759–766. ACM New York, NY, USA, 2007.

[33] M. T. Ribeiro, A. Lacerda, A. Veloso, and N. Ziviani. Pareto-efficient hybridization for multi-objective recommender systems. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 19–26, New York, NY, USA, 2012. ACM.

[34] A. Schwaighofer, V. Tresp, and K. Yu. Learning Gaussian process kernels via hierarchical Bayes. *Advances in Neural Information Processing Systems*, 17:1209–1216, 2005.

[35] G. Shani and A. Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, 2011.

[36] M. Sugiyama, S. Nakajima, H. Kashima, P. von Bunau, and M. Kawanabe. Direct importance estimation with model selection and its application to covariate shift adaptation. *Advances in Neural Information Processing Systems*, 20, 2008.

[37] P. Tseng and S. Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117(1-2):387–423, 2009.

[38] C. Wang and S. Mahadevan. Heterogeneous domain adaptation using manifold alignment. In *IJCAI*, pages 1541–1546, 2011.

[39] Q. Yang, Y. Chen, G.-R. Xue, W. Dai, and Y. Yu. Heterogeneous transfer learning for image clustering via the social web. ACL '09, pages 1–9, 2009.

[40] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen. Improving recommendation lists through topic diversification. In *Proceedings of the 14th international conference on World Wide Web*, pages 22–32. ACM, 2005.