Heteroscedastic Learning and its Realization on Matrix Factorization and Gradient Boosting Regression/Classification

Guang-He Lee
Dept. of Computer Science
National Taiwan University
r04922045@csie.ntu.edu.tw

Shao-Wen Yang
Intel Labs
Intel Corporation
shao-wen.yang@intel.com

Tsung-Hsing Lin
Dept. of Computer Science
National Taiwan University
r05922007@csie.ntu.edu.tw

Shou-De Lin
Dept. of Computer Science
National Taiwan University
sdlin@csie.ntu.edu.tw

ABSTRACT

Given the existence of stochastic noises, the implicit deviations of sample data from their true values are almost surely diverse, which introduces heteroscedasticity and makes each data point not equally suitable for fitting a model. In this case, simply averaging the cost among data in the loss function is not ideal. Intuitively we would like to emphasize more on the reliable instances (i.e., those contain smaller noise) while training a model. Motivated by such observation, we derive a general supervised learning framework which models the uncertainty in addition to the typical label prediction. We instantiate our idea with matrix factorization (MF) and gradient boosting machine (GBM). To our best knowledge, we are among the first to study a low-rank noise structure in MF and to propose a GBM with uncertainty modeling. Our model has two advantages. First, it jointly learns the uncertainty and conducts dynamic reweighting of instances, allowing the model to converge to a better solution. Second, during learning the deviated instances are assigned lower weights, which leads to faster convergence since the model does not need to overfit the noise. The experiments show that our model outperforms the typical empirical risk minimization framework in both accuracy and efficiency in a variety of experiment scenarios including recommendation, missing data imputation, regression, and classification.

CCS CONCEPTS

•Computing methodologies →Supervised learning; Classification and regression trees; Factorization methods;

KEYWORDS

heteroscedasticity, matrix factorization, gradient boosting

ACM Reference format:
Guang-He Lee, Shao-Wen Yang, Tsung-Hsing Lin, and Shou-De Lin. 2017. Heteroscedastic Learning and its Realization on Matrix Factorization and Gradient Boosting Regression/Classification. In *Proceedings of ACM SIGKDD international conference on Knowledge Discovery and Data Mining, Halifax, Nova Scotia Canada, August 2017 (KDD'17),* 9 pages.
DOI: 10.475/123_4

KDD'17, Halifax, Nova Scotia Canada

DOI: 10.475/123_4

1 INTRODUCTION

In general, data can be noisy, and each data point may be contaminated by different level of noise. As long as the noise is non-deterministic, almost surely the sample data will deviate diversely from their expected (noiseless) values. Namely, the existence of noise introduces heteroscedasticity and will cause certain instances (i.e., those with larger noise) to be less trust-able than others. Hence, the typical learning framework under empirical risk minimization, which average the cost among sample data to approximate the expectation of the cost upon data distribution, may not be optimal in this case.

The phenomenon can be characterized in the *unknown*, *ubiquitous*, and *diverse* noises. This is particularly true for sensor data where measurement errors always exist and different sensor has different degree of errors. As the key characteristics is the *diverse importance* among sample data, the same scenario applies to data that are not equally sampled during acquisition, e.g., recommender system [12].

Conventional approaches may not be suitable for this phenomenon. For example, given certain level of trustworthiness for each data point, the instance-weighting strategy is widely adopted [18]. However, such weighting scheme has its limitation since it requires prior knowledge about the trustworthiness of each data point. A related study may be anomaly/outlier detection [37], whose goal is to identify a few (salient) noisy instances from a majority of noiseless data. Our model, on the other hand, assumes the noise is implicitly embedded in every data point, while some contain more and some less.

To address the above issues, we propose a *heteroscedastic learning* framework to model and learn the heteroscedastic deviation, i.e., implicit sample noise, in order to conduct better label prediction. Specifically, we derive the framework from a theoretical model for optimal weighting under heteroscedasticity. By learning the deviation from data, we relieve one less-realistic assumption of the theory that the level of noise is known. Specifically, we proposed a heteroscedastic regression formulation to address the diverse noise

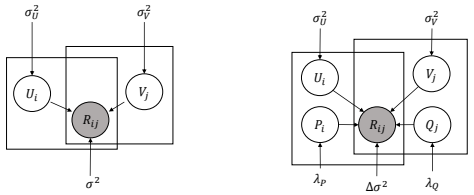

Figure 1: Graphical model representation of probabilistic matrix factorization (PMF) on the left vs. heteroscedastic matrix factorization (HMF) on the right. The factorization P_i, Q_j and $\Delta\sigma^2$ generates heteroscedastic variance $\tilde{\sigma}_{ij}^2$.

phenomenon, and an extension on classification to address the binary characteristics of label noise (label flipping) in classification.

We then realize the heteroscedastic learning framework on matrix factorization (MF) and gradient boosting machine (GBM). In contrast to a traditional sparse noise structure that assumes only limited amount of noise, a novel low-rank structure is adopted in the proposed MF model to model the ubiquitous nature of deviation and to combat overfitting. The illustration of the proposed Heteroscedastic Matrix Factorization (HMF) vs. traditional Probabilistic Matrix Factorization is shown in Figure 1. On the other hand, to our best knowledge, we are among the first to study GBM with uncertainty modeling in the proposed GBM model.

The formulation exhibits two advantages. First, it jointly learns the deviation and conducts dynamic reweighting of instances, allowing the model to converge to a better solution. Second, during learning the deviated instances are assigned lower weights, which leads to faster convergence since the model does not need to overfit the noise.

Comprehensive experiments are conducted in both *noisy* and *clean* data to validate the performance and robustness of proposed heteroscedastic learning method under different scenarios. Specifically, we perform experiments in recommendation, Internet of Things (IoT), regression, and classification datasets. The superior performance of the proposed method to the typical homoscedastic learning framework (empirical risk minimization) is demonstrated in most cases. The implemented code is available at https://guanghelee.github.io/?publication.

2 THE HETEROSCEDASTIC MODEL

2.1 The Ubiquitous Heteroscedasticity

In this work, we first point out that heteroscedasticity in sample data almost surely exist given stochastic noise, even with a homoscedastic noise distribution. We assume each observed data instance (x_i, y_i) comes from the combination of a clean data instance (x_i, y_i^*) and some noise ϵ_i, *sampled* from a continuous random variable ε_i. Besides, the expectation of noise is assumed to be zero. Formally speaking, denoting $\dot{\sim}$ as a sampling process, we have

$$(x_i, y_i = y_i^* + \epsilon_i), \epsilon_i \dot{\sim} \varepsilon_i, E[\varepsilon_i] = 0. \tag{1}$$

For example, in Figure 2, we conduct a random experiment to generate sample data points with $y_i^* = x$ and $\varepsilon_i \sim \mathcal{N}(0, 0.01)$ as circle markers, and correspondingly the denoised data points as square

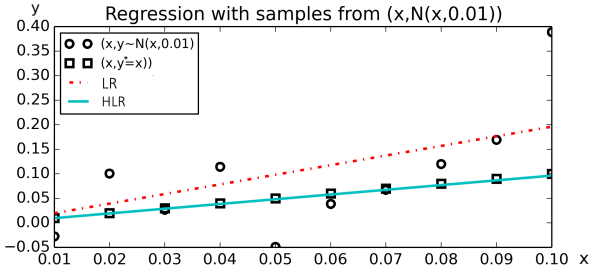

Figure 2: Data points drawn from $(x, y \sim \mathcal{N}(x, 0.01))$ as circle markers. The clean data points of the distribution $(x, y^* = x)$ are denoted as square markers. Despite a homoscedastic variance, the sample data still exhibits heteroscedasticity. In this setting, a heteroscedastic linear regression (HLR) fits the distribution much better than a linear regression (LR) model.

marker. The example shows that even with homoscedastic noise, the sample data still exhibits heteroscedasticity: the deviations from true values are diverse. The phenomenon can also be proven by Lemma 2.1.

Lemma 2.1. *Given* any *two continuous noises ε_i and ε_j with probability density functions $p(x)$ and $q(x)$, it is almost surely that $\varepsilon_i \neq \varepsilon_j$.*

Proof.

$$Pr(\varepsilon_i \neq \varepsilon_j) = \int_{-\infty}^{\infty} p(x)(1 - \int_{x}^{x} q(y)\mathrm{d}y)\mathrm{d}x = \int_{-\infty}^{\infty} p(x)\mathrm{d}x = 1.$$

□

Accordingly, as each data point brings different level of confidence for describing the original distribution, to conduct a fair treatment for data, it is important to consider the diverse deviation for individual data point.

2.2 Weighting for Heteroscedasticity

To address the above issue, an intuitive method is to conduct a weighting scheme for each data point. We begin our derivation from a theoretical framework on a linear model, weighted linear regression (WLR), which solves the following optimization task given weightings W_i for each data point (x_i, y_i):

$$\hat{\omega} = \mathrm{argmin}_{\omega} \frac{1}{N} \sum_{i=1}^{N} \frac{(y_i - \omega^T x_i)^2}{W_i}, \tag{2}$$

where $\hat{\omega}$ is the linear model to be learned. Note that the goal here is only about finding the linear model $\hat{\omega}$ given some weightings $W = \{W_1, ..., W_N\}$, but it remains unknown how to appropriately assign the weightings W.

If data are generated by a linear model ω^* as $y_i = y_i^* + \epsilon_i = (\omega^*)^T x_i + \epsilon_i$, it is ideal to find a weight vector $\hat{\omega}$ that is the closest to the ground truth weight vector ω^* in expectation. That is,

$$\hat{W} = \mathrm{argmin}_{W=\{W_1,\ldots,W_N\}} \mathbb{E}_{\varepsilon_1,\ldots,\varepsilon_N}[(\hat{\omega} - \omega^*)^2]. \tag{3}$$

Note that the weight vector $\hat{\omega}$ is still generated from (2), but the final optimization target is to find the most similar weight vector $\hat{\omega}$ to the ground truth vector ω^* utilizing weightings W.

According to the *generalized Gauss-Markov theorem* [33], the optimal weighting W_i is the variance of noise $Var(\varepsilon_i)$. Since *random variable* ε_i is unobservable in *sample data*, we make an unrealistic assumption that *sample deviation* ϵ_i can be observed. Accordingly, we can use the square of sample deviation ϵ_i^2 to approximate $Var(\varepsilon_i)$ as (4), which we call it the heteroscedastic (weighted) linear regression (HLR).

$$W_i = Var(\varepsilon_i) = E[(\varepsilon_i - E[\varepsilon_i])^2] = E[\epsilon_i^2] \approx \epsilon_i^2. \tag{4}$$

In Figure 2, we compare linear regression (LR) with HLR. The superior performance of HLR is clear for fitting the clean (expected) data. The simple experiment demonstrates the promising efficacy of incorporating weighting for heteroscedasticity, which motivates modeling weighting for heteroscedasticity in a more realistic scenario.

2.3 Heteroscedastic Regression

Since the weight $W_i = \epsilon_i^2$ is unknown in practice, certain strategy to learn the deviation is needed. Here we propose to use a positive estimation $\bar{\sigma}_i^2(\cdot)$ to model the deviation ϵ_i^2

To extend the weighting mechanism in a general regression problem, we models a positive deviation term $\bar{\sigma}_i^2(x_i, \Phi)$ with some parameter Φ in addition to typical label prediction $\bar{y}_i(x_i, \Theta)$ with some parameter Θ. Following the formulation of WLR, we can optimize the parameters Θ, Φ in a weighting formulation as

$$\hat{\Theta}, \hat{\Phi} = \text{argmin}_{\Theta,\Phi} \frac{1}{N} \sum_{i=1}^{N} \frac{(y_i - \bar{y}_i(x_i, \Theta))^2}{\bar{\sigma}_i^2(x_i, \Phi)}, s.t.\ \bar{\sigma}_i^2(x_i, \Phi) > 0. \tag{5}$$

However, the above formulation will reach trivial optimum by simply assigning every $\bar{\sigma}_i^2(x_i, \Phi)$ as infinity. To form a realistic modeling, we perform regularization $\Omega(\cdot)$ on the deviation term as

$$\hat{\Theta}, \hat{\Phi} = \text{argmin}_{\Theta,\Phi} \frac{1}{N} \sum_{i=1}^{N} \frac{(y_i - \bar{y}_i(x_i, \Theta))^2}{\bar{\sigma}_i^2(x_i, \Phi)} + \Omega(\bar{\sigma}_i^2(x_i, \Phi)),$$
$$s.t.\ \bar{\sigma}_i^2(x_i, \Phi) > 0. \tag{6}$$

If adopting natural log as regularization $\Omega(\cdot) = \ln(\cdot)$, the formula (6) will be equivalent to the maximization of a heteroscedastic Gaussian likelihood as,

$$\hat{\Theta}, \hat{\Phi} = \text{argmax} \prod_{i=1}^{N} \frac{\exp(-\frac{(y_i - \bar{y}_i(x_i, \Theta))^2}{2\bar{\sigma}_i^2(x_i, \Phi)})}{\sqrt{2\pi\bar{\sigma}_i^2(x_i, \Phi)}}, s.t.\ \bar{\sigma}_i^2(x_i, \Phi) > 0. \tag{7}$$

The regularization term $\Omega(\bar{\sigma}_i^2(x_i, \Phi)) = \ln(\bar{\sigma}_i^2)$ in (6) corresponds to the $\bar{\sigma}_i^2(x_i, \Phi)$ in the denominator ($\sqrt{2\pi\bar{\sigma}_i^2(x_i, \Phi)}$).

The representation provides a physical meaning of deviation as the variance in a heteroscedastic Gaussian likelihood. Contrasting most conventional modeling for Gaussian likelihood with *shared* variance [1, 27, 30], or the equivalently mean squared error (MSE) minimization, our model addresses the ubiquitous heteroscedasticity.

2.4 Heteroscedastic Classification

As the idea of weighting in the heteroscedastic regression is general, we also extend the idea to a binary classification problem ($y_i \in \{-1, 1\}$) using logistic regression [35], which converts a scalar output $\bar{y}_i \in \mathbb{R}$ to a probability estimation using sigmoid function $\sigma(\cdot)$:

$$Pr(y_i = 1 \mid x_i, \Theta) = \sigma(\bar{y}_i(x_i, \Theta)) = \frac{1}{1 + \exp(-\bar{y}_i(x_i, \Theta))}. \tag{8}$$

The loss function of logistic regression is to minimize the negative log likelihood of data as

$$\frac{1}{N} \sum_{i=1}^{N} \ln(1 + \exp(-y_i \bar{y}_i(x_i, \Theta))). \tag{9}$$

In order to incorporate heteroscedastic modeling, we can also exploit the deviation estimation $\bar{\sigma}_i^2(\cdot)$ in the loss function. However, as the label y_i is restricted to $\{-1, 1\}$, rather than a real value, in binary classification, the original unrestricted positive weighting schema $\bar{\sigma}_i^2(\cdot)$ may not be suitable in this case. Hence, we design another form of deviation estimation that can address the characteristics of label noise in classification. Specifically, in the classification scenario, the noise would either exist as a flipping of the label or not exist at all. Hence, it is desirable to generate a binary deviation weighting to treat each data point as either noisy or clean. As a result, we use another sigmoid function, with estimated logit $d(x_i, \Phi) \in \mathbb{R}$, as a soft approximation to the binary weighting. The final loss can be written as (10).

$$\hat{\Theta}, \hat{\Phi} = \text{argmin}_{\Theta,\Phi} \frac{1}{N} \sum_{i=1}^{N} \frac{\ln(1 + \exp(-y_i \bar{y}_i(x_i, \Theta)))}{\alpha\sigma(d(x_i, \Phi)) + \Delta d} + \Omega(d(x_i, \Phi)), \tag{10}$$

where α is a hyper-parameter to control the maximum weighting when the sigmoid value reaches 1, and Δd is a constant to prevent infinity loss value when the sigmoid value reaches 0.

In sum, we create two novel heteroscedastic learning formulation that supports regression and classification in typical supervised learning scenario. In addition, we address the ubiquitous heteroscedasticity in the heteroscedastic regression formulation, and accommodate the binary characteristics of noise in the heteroscedastic classification formulation. We will instantiate the proposed loss function with concrete models in the next section.

3 CASE STUDY

In this section, we derive concrete models using matrix factorization (MF) and gradient boosting machine (GBM) for heteroscedastic learning. In each case, we will briefly introduce the model and the major contribution of heteroscedastic modeling on each model before proposing a heteroscedastic formulation.

3.1 Case Study 1: Heteroscedastic Matrix Factorization

MF is arguably the most dominant model in collaborative filtering problems that achieve major success on recommender systems [6]. However, [12] points out that the distribution of user activity is very skewed: very few users would rate lots of items, and vice versa. This phenomenon implies different level of importance of each rating record to account for the latent preference of each user, which echos the notion of heteroscedastic learning to address the unknown and diverse weighting.

Formally speaking, given a matrix $R \in \mathbb{R}^{N_1 x N_2}$, MF reconstruct it as matrix $\bar{R} \in \mathbb{R}^{N_1 x N_2}$ by low rank matrices $U \in \mathbb{R}^{N_1 x D}$ and $V \in \mathbb{R}^{N_2 x D}$ with latent dimension D. A widely used variant [6, 18] is to add some bias and scalar terms to the factorization as follows:

$$\bar{R}_{ij} = U_i^T V_j + u_i + v_j + \mu. \tag{11}$$

u_i and v_j are biases, which are scalars to be learned, for user i and item j, respectively. μ is the average value for observed elements of R in the training data. In the remainder of the paper, u_i, v_j, and μ are omitted for clarity purpose.

3.1.1 Heteroscedastic Modeling. Here we propose the idea of Heteroscedastic Matrix Factorization (HMF). Conventionally, the data can be modeled by a homoscedastic Gaussian likelihood with mean constructed by MF. To adapt MF to a heteroscedastic model, we have to further model the variance in the Gaussian likelihood.

The key concern lies in the choice of variance structure. Given the ubiquitous nature of noise, a sparse structure might not be ideal and it is preferable to build a prediction model for every element on the matrix. Therefore, a low-rank structure to represent such dense variance matrix is desirable. In addition, the low-rank structure also regularize the model complexity from overfitting. Consequently, we can adopt another MF to construct the variance $\bar{\sigma}_{ij}^2$ by deviation latent factors, P_i and Q_j.

Since variance has a non-negative constraint, we can impose a non-negative prior on deviation latent factors, thus the inner product of non-negative vectors is also non-negative. In addition, to avoid a zero value of $\bar{\sigma}_{ij}^2$ that leads to infinite loss value, we add a small positive value $\Delta\sigma^2$ as (12).

$$\bar{\sigma}_{ij}^2 = P_i^T Q_j + \Delta\sigma^2. \tag{12}$$

To impose a non-negative prior, we select exponential distribution, which encourages sparse solution for efficiency [11]. Accordingly, the prior distribution on deviation latent factor results in (13).

$$p(P_i|\lambda_P) = \mathcal{EXP}(P_i|\lambda_P I),\ p(Q_j|\lambda_Q) = \mathcal{EXP}(Q_j|\lambda_Q I). \tag{13}$$

With Gaussian prior on mean latent factors, the mean prior and likelihood of HMF can be represented as (14) and (15), with I_{ij} being a binary indicator of the availability of R_{ij} in the training data.

$$p(U_i|\sigma_U^2) = \mathcal{N}(U_i|0, \sigma_U^2 I),\ p(V_j|\sigma_V^2) = \mathcal{N}(V_j|0, \sigma_V^2 I), \tag{14}$$

$$p(R_{ij}|U_i, V_j, P_i, Q_j) = \mathcal{N}(R_{ij}|\bar{R}_{ij}, \bar{\sigma}_{ij}^2)^{I_{ij}}. \tag{15}$$

The log posterior distribution of HMF can be derived as the following form,

$$\begin{aligned}
&-\sum_{i=1}^{N_1}\sum_{j=1}^{N_2}\frac{I_{ij}(R_{ij}-U_i^T V_j)^2}{2(P_i^T Q_j+\Delta\sigma^2)} - \sum_{i=1}^{N_1}(\frac{U_i^T U_i}{2\sigma_U^2} + \lambda_P P_i^T I)\\
&-\sum_{j=1}^{N_2}(\frac{V_j^T V_j}{2\sigma_V^2} + \lambda_Q Q_j^T I) - \frac{D}{2}N_1 ln(\sigma_U^2)\\
&-\frac{D}{2}N_2 ln(\sigma_V^2) + DN_1 ln(\lambda_P) + DN_2 ln(\lambda_Q)\\
&-\frac{1}{2}\sum_{i=1}^{N_1}\sum_{j=1}^{N_2} I_{ij}\ln(P_i^T Q_j+\Delta\sigma^2) + const,\\
&\text{subject to } P_{ik}, Q_{jk} \geq 0, i \in [1, N_1], j \in [1, N_2].
\end{aligned} \tag{16}$$

3.1.2 Learning Heteroscedastic Matrix Factorization. To learn HMF, one plausible strategy is to resort to stochastic gradient ascent (SG) for maximizing the log posterior distribution (MAP) since the objective function is differentiable. However, with the non-negative constraint on deviation latent factors, learning P and Q is not trivial. The constraint results in non-negative MF with the deviation latent factors. Fortunately, there have been several methods proposed for such constraint [24], while we exploit a simple yet effective method called the projected gradient method. That is, every time a single step SG is done, all negative results in deviation latent factors are replaced with 0.

The SG learning of HMF is linear to the number of observed data, i.e., O(# of epochs × # of observed data × D). Moreover, the space complexity is O($(N_1 + N_2) \times$ D). Both time and space complexity are the same as the conventional MF model. In our experiment, the deviation latent factors achieve at least 33% sparsity through exponential prior.

3.2 Case Study 2: Heteroscedastic Gradient Boosting Machine

Tree-based models, such as random forest [23], are effective in practical data mining competitions [39]. However, as the node splitting stage in decision tree [4] training usually requires analytical solution for model score in each split to efficiently search for the optimal split, it is hard to derive a joint terminal split strategy with analytical solutions for both uncertainty weight assignment and label prediction.

The major contribution in this case is that the prosed algorithm, heteroscedastic gradient boosting machine (HGBM), can learn uncertainty measurement and label prediction simultaneously with analytical solution during terminal split search. HGBM is derived based on a popular variant of GBM, XGBoost, which exploited by 29 Kaggle's competition winners in 2015 [7].

Boosting is a class of learning algorithm that additively builds a (sub-)model, which is typically a decision tree [9], upon all historical models. Therefore, in T^{th} iteration, the optimization is *only* conducted on the new model $f_T(\cdot)$, where the label prediction $\bar{y}$ is written as the addition of new model $f_T(\cdot)$ and historical models $\sum_{t=1}^{T-1} f_t(\cdot)$ as

$$\bar{y}_i(x_i, \Theta) = \sum_{t=1}^{T} f_t(x_i), \tag{17}$$

where Θ consists of parameters in $F_T = \{f_1, ..., f_T\}$. Accordingly, given $T-1(\geq 0)$ established trees, GBM derive gradient on the loss function with respect to $f_T(\cdot)$, and use a new regression tree to fit the gradient value. More details in the original formulation can be referred to [9].

3.2.1 Heteroscedastic Modeling. As the framework under GBM only models a label prediction, the framework does not support our heteroscedastic modeling where label prediction and deviation estimation are jointly learned. In order to extend our model to the gradient boosting framework,we establish two distinct models, $f_t(\cdot)$ and $\bar{f}_t(\cdot)$, for label prediction and deviation estimation, respectively. Accordingly, the deviation estimation is constructed by another set

of boosting trees as

$$d_i(x_i, \Phi) = \sum_{t=1}^{T} \bar{f}_t(x_i). \tag{18}$$

The formulation for $\bar{\sigma}_i^2(x_i, \Phi)$ is similar[1]. As both (6) and (10) are differentiable functions, we simplify the notation by using $\mathcal{L}(\{x_i, y_i\}, F_T, \bar{F}_T)$, where $\bar{F}_T$ denotes $\{\bar{f}_1, ..., \bar{f}_T\}$, to denote a differentiable weighted cost function as in (6) or (10). In addition, to incorporate regularization for both function values, F_T and $\bar{F}_T$, we abuse a single notation $\Omega(F_t(x_i), \bar{F}_t(x_i))$ to denote the regularization on data x_i. The final loss function can be simplified as

$$\sum_{i=1}^{N} \mathcal{L}(\{x_i, y_i\}, F_T, \bar{F}_T) + \Omega(F_t(x_i), \bar{F}_t(x_i)). \tag{19}$$

We conduct second order Taylor approximation, which can be further used to derive analytical solution for model scores [7], of the weighted cost at $f_t(x_i) = 0$ and $\bar{f}_t(x_i) = 0$:

$$\sum_{i=1}^{N} \mathcal{L}(\{x_i, y_i\}, F_T, \bar{F}_T) \approx \sum_{i=1}^{N} [\mathcal{L}(\{x_i, y_i\}, F_{T-1}, \bar{F}_{T-1}) + g_i f_t(x_i) + \bar{g}_i \bar{f}_t(x_i) + \frac{1}{2}(h_i f_t(x_i)^2 + 2\tilde{h_i} f_t(x_i) \bar{f}_t(x_i) + \bar{h}_i \bar{f}_t(x_i)^2)], \tag{20}$$

where $g_i, \bar{g}_i$ are first order gradients to $\mathcal{L}(\cdot)$, while $h_i, \tilde{h_i}, \bar{h}_i$ are second order gradients. Since the optimization is only targeted on the current function f_t and $\bar{f}_t$, $\mathcal{L}(\{x_i, y_i\}, F_{T-1}, \bar{F}_{T-1})$ is thus a constant that does not affect optimization. Note that different from typical one dimensional Taylor approximation in XGBoost [7], (20) involves two dimension.

On the other hand, as the (sub-)models, f_T and $\bar{f}_T$, are decision trees, a typical choice of regularization is conducted on the number of terminal nodes, K and $\bar{K}$, and model score in each terminal node w_j and $\bar{w}_j$ for respective trees as

$$\sum_{i=1}^{N} \Omega(F_t(x_i), \bar{F}_t(x_i)) = \gamma K + \bar{\gamma} \bar{K} + \frac{\lambda}{2} \sum_{j=1}^{K} w_j^2 + \frac{\bar{\lambda}}{2} \sum_{j=1}^{\bar{K}} \bar{w}_j^2, \tag{21}$$

where $\gamma, \bar{\gamma}, \lambda, \bar{\lambda}$ are hyper-parameters and $w_j, \bar{w}_j$ are the model score in the j^{th} terminal node for respective trees, $f_T(\cdot), \bar{f}_T(\cdot)$. Note that we omitted the iteration index T in the model scores, w_j and $\bar{w}_j$, and number of terminal nodes, K and $\bar{K}$, for clarity purpose.

3.2.2 Learning Heteroscedastic Gradient Boosting Machine. As $f_t(x_i)$ is a mapping function from data instance x_i to the model score w_j in some tree terminal node j, we can compute the sum of gradients among data instances within each tree nodes j to formulate a more organized loss function. If we denote the set of data instance in a terminal j at the T^{th} iteration as $I_T(j)$ for the label prediction and as $\bar{I}_T(j)$ for the deviation estimation, the loss function (19) can be rewritten as

$$\sum_{j=1}^{K} [\sum_{i \in I_T(j)} g_i w_j + \frac{1}{2}(\sum_{i \in I_T(j)} h_i + \lambda) w_j^2] + \sum_{i=1}^{N} \tilde{h_i} f_t(x_i) \bar{f}_t(x_i) + \sum_{j=1}^{\bar{K}} [\sum_{i \in \bar{I}_T(j)} \bar{g}_i \bar{w}_j + \frac{1}{2}(\sum_{i \in \bar{I}_T(j)} \bar{h}_i + \bar{\lambda}) \bar{w}_j^2] + \gamma K + \bar{\gamma} \bar{K}. \tag{22}$$

As the optimal model score for each terminal nodes, e.g., w_j, become analytical if we omit one of the second order gradients, $\tilde{h_i} f_t(x_i) \bar{f}_t(x_i)$, we further simplified the loss function in this way to derive analytical model score for each terminal node j:

$$w_j = \frac{-\sum_{i \in I_T(j)} g_i}{\sum_{i \in I_T(j)} h_i + \lambda}; \quad \bar{w}_j = \frac{-\sum_{i \in \bar{I}_T(j)} \bar{g}_i}{\sum_{i \in \bar{I}_T(j)} \bar{h}_i + \bar{\lambda}} \tag{23}$$

Finally, in each tree terminal, we search for all possible splits that would optimize the loss function. For example, at terminal node j in the label estimation tree, we pick the split u and v ($I_t(u) \cup I_t(v) = I_t(j), I_t(u) \cap I_t(v) = \varnothing$) with positive and maximal loss decrease among all possible splits. The loss decrease can be derived as

$$\frac{1}{2}[\frac{(\sum_{i \in I_T(j)} g_i)^2}{\sum_{i \in I_T(j)} h_i + \lambda} - \frac{(\sum_{i \in I_T(u)} g_i)^2}{\sum_{i \in I_T(u)} h_i + \lambda} - \frac{(\sum_{i \in I_T(v)} g_i)^2}{\sum_{i \in I_T(v)} h_i + \lambda}] + \gamma, ^2 \tag{24}$$

where the term γ is introduced from (21) to regularize on the number of terminal nodes, which exhibits an automatic *pruning* mechanism [7]: a terminal node split is invalid if the objective value improvement within the brackets in (24) is less than γ, the split is invalid. The derivation for deviation estimation tree is similar and omitted for space limitation.

The training process of HGBM is simply an iteration process to establish a label prediction tree $f_T(\cdot)$ and a deviation estimation tree $\bar{f}_T(\cdot)$ in each iteration T based on (24).

4 DISCUSSION

In this section, we provide some insights of heteroscedastic learning to distinguish its roles in the learning process.

4.1 Heteroscedastic Learning as Dynamic Weight Scheduling

In literature, several methods have been proposed for the per-coordinate learning rate scheduling on first order gradient descent learning [8, 29]. Our model treats weight scheduling from different angle: our model essentially provides dynamic learning weights on the cost, e.g., $(y_i - \bar{y}_i)^2$, for different instances y_i during optimization, as shown in (6).

Heteroscedastic learning can also be viewed as a regularization scheme for different instances. As deviation controls the fitting for data, it yields a looser constraint of fitting for less informative instance with higher deviation, and vice versa. Accordingly, faster convergence can be expected as it does not need to overfit the noisy data (i.e., those with high deviation). Besides, contrary to the conventional regularization scheme that manipulates regularization on model parameters to control the fitting on data, heteroscedastic

[1] We find that using $d_i(x_i, \Phi)$ is superior to $\bar{\sigma}_i^2(x_i, \Phi)$ for HGBM in both classification and regression experiments, and thus will derive HGBM using $d_i(x_i, \Phi)$ for simplicity.

[2] It is possible to derive a single split strategy for both label prediction and deviation estimation within a single tree, but the proposed separate split strategies for $f_t(\cdot)$ and $\bar{f}_t(\cdot)$ are more flexible to allow different model complexity for label prediction $f_t(\cdot)$ and deviation estimation $\bar{f}_t(\cdot)$.

learning directly regularize the fitting for noisy data by deviation estimation.

Also note that the deviation of our model is learned jointly with the label prediction, so deviation can dynamically coordinate with label prediction throughout the whole optimization process. Thus, in each iteration, the deviation can be adjusted and immediately used to re-weight the cost function to learn a better prediction model. In contrast, in traditional weighting method [18], the confidence level of data point is assumed to be explicitly given, or learned separately with the model.

4.2 Heteroscedastic Learning as a General Framework

Our model assumes the existence of ubiquitous and diverse noise, i.e. heteroscedasticity. Hence, it is worthy to consider when to adopt such technique. First, if the noises implicitly or explicitly exist, it is clear our model is useful. Second, it applies to situation when the data instances shall not be treated equally in model training (e.g. due to biased sampling of instances). Finally, given perfectly clean data, our model also allows a solution with shared or near zero deviation values. As a result, given insufficient knowledge about whether data are noisy and sufficient amount of training data, heteroscedastic learning is a favorable choice than the conventional homoscedastic formulation, which is validated in the next section.

5 EXPERIMENTS

We conduct empirical study to evaluate our model in a variety of experiment environments, including recommender system (RS), Internet of Things (IoT), classic classification and regression datasets. The detailed parameter setting in each experiment can be found on our material website.

5.1 HMF for Recommendation

Experiments are conducted on three standard movie recommendation datasets: MovieLens1M, MovieLens10M, and Netflix dataset. Each dataset can be represented by a user vs. movie matrix R. The statistics is shown on Table 1.

The experiments are conducted using the protocol proposed in the state-of-the-art solutions [21, 32]. Specifically, we produce a random 90% and 10% split of data for training and testing, and repeat 5 times to generate 5 different splits. Among the training data, 10% of them are used as validation set for parameter tuning. For testing data without user or movie observed in training, a default rating 3 is predicted for convention [32]. We report the average root mean squared error (RMSE) on testing data.

We compare HMF with basic and state-of-the-art collaborative filtering (CF) models:

- *Restricted Boltzmann Machines (RBM)* [6]. A widely used generative model for CF.
- *Biased Matrix Factorization (Biased MF)*. A simple yet successful method for recommendation as introduced in (11), the homoscedastic counterpart of HMF.
- *Robust Collaborative Filtering (RCF)* [25]. A robust weighted MF model applying Huber M-estimator [14] to iteratively estimate the weighting during optimization. L_2 regularization is added for better generalization performance.
- *Robust Bayesian Matrix Factorization (RBMF)* [20]. A robust MF adopting a rank-1 structure for noise and Student-t prior on mean latent factors.
- *Local Low-Rank Matrix Approximation (LLORMA)* [21]. The state-of-the-art matrix approximation model assuming the target matrix is locally low-rank.
- *AutoRec* [32]. The state-of-the-art deep learning recommendation model based on Auto-Encoder.

We also conduct experiments for the state-of-the-art anomaly detection method, Direct Robust MF (DRMF) [37], based on Biased MF. However, the training and testing performance of DRMF decrease at almost every iteration in the anomaly detection process, so the best performance in DRMF is equivalent to Biased MF.

The results on Table 2 show that HMF outperforms all the state-of-the-art in MovieLens10M and Netflix by a large margin. The only exception comes from the smaller dataset (i.e., MovieLens1M) where our model might have overfitted the training data due to insufficient data.

Since HMF, Biased MF, RCF, and RBMF share similar search space for label prediction, it is worthy to compare the these MF models. First, HMF significantly outperforms Biased MF, its homoscedastic counterpart, in all datasets. Since the three datasets are not recognized as noisy in literature, the phenomenon may imply potential implicit noise or uneven sampling among users/items as we hypothesized. It demonstrates that jointly learning deviation and label prediction yields more accurate prediction. For RCF, using an M-estimator to estimate the weight is even worse than vanilla Biased MF. Compared with RBMF, HMF is superior in larger datasets, demonstrating the efficacy of low-rank (e.g., 100) variance structure versus the rank-1 variance structure in RBMF.

Finally, compared with nonlinear AutoRec model, linear HMF is also superior. We argue the phenomenon comes from the sparsity of real world data that a nonlinear model is prone to overfit. Similar scenario appears in knowledge base completion that a linear model is reported to dominate non-linear models [38].

5.2 Missing Sensor Data Imputation on IoT data

The task of missing sensor data imputation is to infer missing testing data given observed training data. The experiments are conducted on two real world IoT datasets, an outdoor air quality monitoring dataset from London and an indoor sensor dataset from Intel Berkeley Research lab, which is open to public[3]. The IoT data are naturally noisy due to sensor measurement errors. Each dataset can be represented by a 3-mode tensor T: the three dimensions denote location, timestamp, and sensor type, respectively. Note that there are multiple types of sensors (e.g., humidity, light and heat sensors) in each location. The statistics are shown in the right of Table 1. Since multi-modal sensors may have different ranges of values, data are normalized for each type of sensor by standard score.

We make a generalization of HMF to Heteroscedastic Tensor Factorization (HTF) by replacing the MF models in HMF with 3-mode PARAFAC tensor factorization [36] models for label prediction and deviation estimation. The time complexity of HTF is trivially 1.5

[3] http://db.csail.mit.edu/labdata/labdata.html

Table 1: Statistics of recommender system (RS) and Internet of Things (IoT) Datasets

RS	MovieLens1M	MovieLens10M	Netflix	IoT	London	Berkeley
# of users	6,040	69,878	480,189	# of locations	15	58
# of movies	3,706	10,677	17,770	# of sensors	19	4
-	-	-	-	# of timestamps	7,479	46,613
# of data	1M	10M	100M	# of data	476,638	9,158,515
sparsity	95.53%	98.66%	98.82%	sparsity	77.64%	15.31%

Table 2: Performance in RS: all RMSE values have 95% confidence intervals $\leq \pm 0.003$ except RCF (± 0.005).

	MovieLens1M	MovieLens10M	Netflix
RBM	0.881	0.823	0.845
Biased MF	0.845	0.803	0.844
RCF	0.867	0.809	0.855
RBMF	0.837	0.795	0.824
LLORMA	0.833	0.782	0.834
AutoRec	**0.831**	0.782	0.823
HMF	0.841	**0.775**	**0.806**

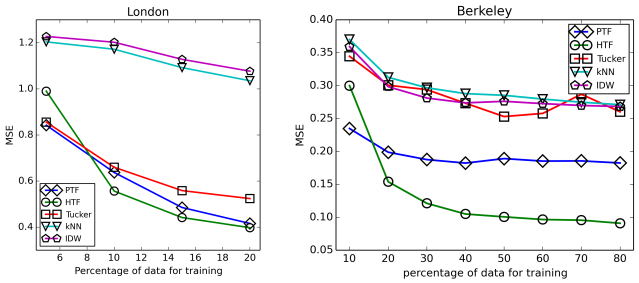

Figure 3: Prediction error over various percentage of training data in London and Berkeley IoT datasets. The range of x-axis may vary according to the original sparsity of dataset. For example, we only have 22.36% of data available in London data, so we conduct experiments using {5%, 10%, 15%, 20%} of training data.

times of HMF. All experiments are repeated 100 times, and the average result is reported. Here we exploit mean squared error (MSE) as our evaluation metric. HTF is compared with the following classic sensor data imputation and factorization models:

- *Spatial k-Nearest Neighbors (kNN).* Average of K spatially nearest data points as prediction.
- *Inversed Distance Weighting (IDW).* Using the inverse of distance for data points as the weighting for interpolation.
- *Probabilistic Tensor Factorization (PTF)* [36]. PTF is equivalent to PARAFAC tensor factorization with L_2 norm regularization, the homoscedastic counterpart of HTF.
- *Tucker* [17]. Tucker Decomposition with L_2 regularization added for better performance.

Table 3: Statistics of classification and regression datasets.

classification datasets			*regression datasets*		
dataset	data #	feature #	dataset	data #	feature #
breast cancer	277	9	abalone	4,177	8
diabetes	768	8	housing	506	13
thyroid	215	5	cpusmall	8,192	12
german	1,000	20	bodyfat	252	14
heart	270	13	pyrim	74	27

The experiment are conducted through various percentages of data as training data. The setting for each dataset may vary according to its original sparsity. Results are shown in Figure 3. It can be noticed that except the most sparse setting in each dataset, our method outperforms all competitors significantly. A reasonable explanation is that when data are already sparse, the downgrading of certain potentially noisy data can lead to insufficient amount of information for training. For winning cases, we also conduct unpaired t-test with the competitors and confirmed that all P values are smaller than 0.0001. In summary, this section demonstrates the efficacy of heteroscedastic learning under different amount of training data over the homoscedastic methods.

5.3 Classification and Regression

This section shows as a case study to demonstrate the efficacy of heteroscedastic learning for GBM, we conduct experiments using the UCI classification datasets[4] and regression datasets from LIBSVM datasets[5]. The detail statistics of the datasets is available on Table 3. For each classification and regression dataset, we prepare 100 different splits for training and testing. The preprocessed 100 training/testing splits in UCI classification datasets are used directly, and a random 80% and 20% split of data are used for training and testing in each regression dataset. For parameter tuning, 20% of the training data are used as validation. We report the average accuracy for classification and average mean squared error (MSE) for regression among all splits.

We compare HGBM with its homoscedastic counterpart, GBM, to test the efficacy of introducing heteroscedasticity among various datatsets. Especially, we performance experiments in both clean setting and noisy setting to test the robustness of heteroscedastic learning. In the noisy setting, 20% of the positive and negative training label in classification datasets are flipped, whereas in regression

[4]Preprocessed dataset downloaded from http://theoval.cmp.uea.ac.uk/matlab/
[5]https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

Table 4: Performances in classification datasets in terms of accuracy, and regression datasets in terms of mean squared error (MSE). * denotes statistical significance with P value < 0.5.

classification datasets			*regression datasets*		
dataset	GBM	HGBM	dataset	GBM	HGBM
clean training data					
breast cancer	**0.7379**	0.7277	abalone	5.0808	**4.9775**
diabetes	0.7471	**0.7518**	housing	10.4650	**9.9973**
thyroid	**0.9318**	0.9238	cpusmall	7.7789	**7.7758**
german	0.7085	**0.7580***	bodyfat	0.000099	**0.000067***
heart	**0.8051**	0.7965	pyrim	0.1093	**0.0117**
20% training data are added artificial noise					
breast cancer	0.6654	**0.7161***	abalone	4.8219	**4.7515**
diabetes	**0.7362**	0.7332	housing	10.2646	**10.1911**
thyroid	0.8471	**0.8750***	cpusmall	7.8407	**7.5924***
german	0.6973	**0.7234***	bodyfat	0.000099	**0.000067***
heart	0.6744	**0.7408***	pyrim	0.0483	**0.0114**

Table 5: Running time analysis (in seconds) in MovieLens1M datasets

	Biased MF	AutoRec	HMF
Total running time	207	647	88
Running time/epoch	2.1	3.3	4.4

20% of the training label are added a Gaussian noise $\mathcal{N}(0, 0.1\hat{\sigma}^2)$. The variance of the Gaussian distribution $\hat{\sigma}^2$ is calculated as the variance in all training labels. Note that we only add the noises in the training data; the testing data remain clean.

The experiment results with statistical significance are shown on Table 4. The proposed HGBM outperforms GBM in almost every dataset in the noisy setting. On the other hand, HGBM remains competitive in the clean setting, winning 7 out of 10 experiment datasets. Through experiments in a variety of datasets, the promising result not only shows the superiority of heteroscedastic learning in a noisy setting, but also the robustness of proposed heteroscedastic learning in a clean setting.

5.4 Running Time and Convergence

We also compare the running time in 1 million MovieLens1M dataset with the strongest competitor, AutoRec, and the homoscedastic counterpart of HMF, Biased MF. All above models are implemented in C++. AdaGrad optimizer [8] is used for HMF and Biased MF, and resilient propagation optimizer [29] is used for AutoRec, as suggested in the original paper [32]. The running time for each model, with early stopping tuned by validation, is shown in Table 5. Though the running time per epoch of HMF is the longest, it takes much fewer epochs to converge and in turn spent the least amount of total running time.

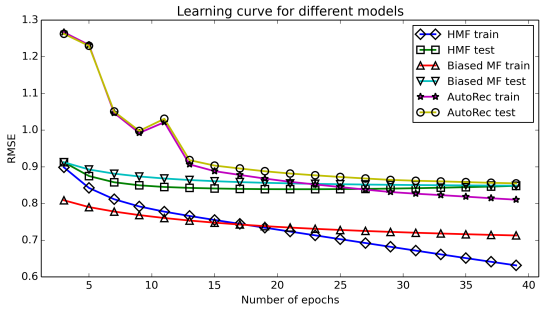

Figure 4: Learning curves in MovieLens1M.

We also conduct learning curve comparison of HMF with Biased MF and AutoRec. The results are shown in Figure 4. Comparing Biased MF with HMF, it confirms our hypothesis that by downplaying the importance of data with larger deviation (i.e., noisy data), our model can converge faster than Biased MF. The fast convergence may imply early overfitting, but the generalization performance of heteroscedastic learning remains competitive according to previous experiments.

6 RELATED WORKS

In traditional weighting method [18], the confidence level of data point is assumed to be explicitly given, or learned separately with the model. Cost-sensitive learning [40] may also be a relevant research area in a sense that different weight (cost) is associated with each data (label) instance. However, we focus on a scenario that such weighting is not available. There are also some orthogonal works in weighting data proposing to organize, rather than randomly present as SG, data points in a meaningful manner for training a better non-convex model [2, 19]. Since the proposed heteroscedastic learning does not alter the typical optimization process, it can be coordinated with [2, 19]. In the literature of collaborative filtering, [13] and [22] are designed for an implicit feedback scenario by a weighting mechanism, while our method addresses an explicit feedback scenario. [31] aims at weighting the distribution of data to get unbiased estimation, but we argue that real world data, like MovieLens1M and Netflix, are biased and thus should use biased data for evaluation.

There are lots of works developed for specific models to be robust to noise, such as principal component analysis (PCA) [41], linear regression [34], logistic regression [35], adaptive boosting (AdaBoost) [3], and deep learning [15, 16]. In contrast, we derive a heteroscedastic learning formulation for general supervised learning problems. Similar to our solution, [10, 28] propose a solution for empirical risk minimization. However, [28] is only applicable to convex surrogates, whereas our solution supports non-convex models as MF. Similarly, [10] only derives linear model with their theoretical framework, in contrast with the proposed tree-based HGBM. Finally, to our best knowledge, HGBM is the first robust GBM model in the literature.

In related works for robust MF, some works are designed for factorization on full matrix rather than CF. For example, [26] and [5] aim at recovering clean image. For robust CF, RCF [25] proposes to use Huber M-estimator [14] to estimate the weighting,

compared with our learning strategy. RBMF [20] is the most related one, which also models heteroscedastic variance; however, RBMF adopts a rank-1 variance structure, rather than our low-rank structure. In addition, the formulation of RBMF is intractable, requiring variational inference method to derive a tractable lower-bound, while HMF can be trained directly by SG.

Finally, a great portion of noise-aware models adopt a sparse structure for noise [37], contrary to our low-rank structure in HMF. The discrepancy stems from the underlying assumption. A sparse noise structure deals with only a few (salient) noises, while we assume noise is everywhere. Moreover, low-rank structure is an efficient methodology to avoid overfitting the noises.

7 CONCLUSIONS

This paper proposes a heteroscedastic objective function that considers diverse weights for each instance while learning a model. The main challenge lies in the fact that the deviation, or noise level, of each instance is unknown, which leads to the major contribution of this paper to propose a joint learning framework that can dynamically estimate the deviation and utilize it to adjust the cost function during training. Note that the recommendation datasets we used are all not known as noisy, which implies that even when data are clean, modeling the deviation as proposed can still yield superior performance, maybe due to the fact that data are actually implicitly noisy or not equally sampled during acquisition.

REFERENCES

[1] Deepak Agarwal and Bee-Chung Chen. 2010. fLDA: matrix factorization through latent dirichlet allocation. In *Proceedings of the third ACM international conference on Web search and data mining*. ACM, 91–100.

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*. ACM, 41–48.

[3] Jakramate Bootkrajang and Ata Kabán. 2013. Boosting in the presence of label noise. *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence* (2013).

[4] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. 1984. *Classification and regression trees*. CRC press.

[5] Xiangyong Cao, Yang Chen, Qian Zhao, Deyu Meng, Yao Wang, Dong Wang, and Zongben Xu. 2015. Low-rank matrix factorization under general mixture noise distributions. In *Proceedings of the IEEE ICCV*. 1493–1501.

[6] Po-Lung Chen, Chen-Tse Tsai, Yao-Nan Chen, Ku-Chun Chou, Chun-Liang Li, Cheng-Hao Tsai, Kuan-Wei Wu, Yu-Cheng Chou, Chung-Yi Li, Wei-Shih Lin, and others. 2011. A linear ensemble of individual and blended models for music rating prediction. *KDDCup* (2011).

[7] 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*. ACM, 785–794.

[8] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research* 12 (2011), 2121–2159.

[9] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[10] Wei Gao, Lu Wang, Yu-Feng Li, and Zhi-Hua Zhou. 2016. Risk Minimization in the Presence of Label Noise.. In *AAAI*. 1575–1581.

[11] Prem Gopalan, Jake M Hofman, and David M Blei. 2013. Scalable recommendation with poisson factorization. *arXiv preprint arXiv:1311.1704* (2013).

[12] Prem Gopalan, Jake M Hofman, and David M Blei. 2015. Scalable Recommendation with Hierarchical Poisson Factorization.. In *UAI*. 326–335.

[13] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*. Ieee, 263–272.

[14] Peter J Huber. 2011. *Robust statistics*. Springer.

[15] Ishan Jindal, Matthew Nokleby, and Xuewen Chen. 2016. Learning Deep Networks from Noisy Labels with Dropout Regularization. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, 967–972.

[16] Pravin Kakar and Alex Yong-Sang Chia. 2015. If you can't beat them, join them: Learning with noisy data. In *Proceedings of the 23rd ACM international conference on Multimedia*. ACM, 571–580.

[17] Tamara G Kolda and Brett W Bader. 2009. Tensor decompositions and applications. *SIAM review* 51, 3 (2009), 455–500.

[18] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 8 (2009), 30–37.

[19] M Pawan Kumar, Benjamin Packer, and Daphne Koller. 2010. Self-paced learning for latent variable models. In *Advances in Neural Information Processing Systems*. 1189–1197.

[20] Balaji Lakshminarayanan, Guillaume Bouchard, and Cedric Archambeau. 2011. Robust Bayesian Matrix Factorisation.. In *AISTATS*. 425–433.

[21] Joonseok Lee, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2013. Local low-rank matrix approximation. In *Proceedings of The 30th International Conference on Machine Learning*. 82–90.

[22] Dawen Liang, Laurent Charlin, James McInerney, and David M Blei. 2016. Modeling user exposure in recommendation. In *Proceedings of the 25th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 951–961.

[23] Andy Liaw and Matthew Wiener. 2002. Classification and regression by randomForest. *R news* 2, 3 (2002), 18–22.

[24] Chih-Jen Lin. 2007. Projected gradient methods for nonnegative matrix factorization. *Neural computation* 19, 10 (2007), 2756–2779.

[25] Bhaskar Mehta, Thomas Hofmann, and Wolfgang Nejdl. 2007. Robust collaborative filtering. In *Proceedings of the 2007 ACM conference on Recommender systems*. ACM, 49–56.

[26] Deyu Meng and Fernando De La Torre. 2013. Robust matrix factorization with unknown noise. In *Proceedings of the IEEE International Conference on Computer Vision*. 1337–1344.

[27] Andriy Mnih and Ruslan Salakhutdinov. 2007. Probabilistic matrix factorization. In *Advances in neural information processing systems*. 1257–1264.

[28] Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. 2013. Learning with noisy labels. In *Advances in neural information processing systems*. 1196–1204.

[29] Martin Riedmiller and Heinrich Braun. 1993. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Neural Networks, 1993., IEEE International Conference on*. IEEE, 586–591.

[30] Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th international conference on Machine learning*. ACM, 880–887.

[31] Tobias Schnabel, Adith Swaminathan, Ashudeep Singh, Navin Chandak, and Thorsten Joachims. 2016. Recommendations as treatments: Debiasing learning and evaluation. *arXiv preprint arXiv:1602.05352* (2016).

[32] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. AutoRec: Autoencoders Meet Collaborative Filtering. In *Proceedings of the 24th International Conference on World Wide Web Companion*. International World Wide Web Conferences Steering Committee, 111–112.

[33] Cosma Shalizi. 2013. Advanced data analysis from an elementary point of view. (2013).

[34] Yiyuan She and Art B Owen. 2011. Outlier detection using nonconvex penalized regression. *J. Amer. Statist. Assoc.* 106, 494 (2011), 626–639.

[35] Julie Tibshirani and Christopher D Manning. 2014. Robust Logistic Regression using Shift Parameters.. In *ACL*. 124–129.

[36] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff G Schneider, and Jaime G Carbonell. 2010. Temporal Collaborative Filtering with Bayesian Probabilistic Tensor Factorization.. In *SDM*, Vol. 10. SIAM, 211–222.

[37] Liang Xiong, Xi Chen, and Jeff Schneider. 2011. Direct robust matrix factorizatoin for anomaly detection. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on*. IEEE, 844–853.

[38] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2014. Embedding entities and relations for learning and inference in knowledge bases. *arXiv preprint arXiv:1412.6575* (2014).

[39] Hsiang-Fu Yu, Hung-Yi Lo, Hsun-Ping Hsieh, Jing-Kai Lou, Todd G McKenzie, Jung-Wei Chou, Po-Han Chung, Chia-Hua Ho, Chun-Fu Chang, Yin-Hsuan Wei, and others. 2010. Feature engineering and classifier ensemble for KDD cup 2010. In *KDD Cup*.

[40] Bianca Zadrozny, John Langford, and Naoki Abe. 2003. Cost-sensitive learning by cost-proportionate example weighting. In *Data Mining, 2003. ICDM 2003. Third IEEE International Conference on*. IEEE, 435–442.

[41] Qian Zhao, Deyu Meng, Zongben Xu, Wangmeng Zuo, and Lei Zhang. 2014. Robust Principal Component Analysis with Complex Noise.. In *ICML*. 55–63.