Mobile App Risk Ranking via Exclusive Sparse Coding[*]

Deguang Kong
Yahoo Research
701 First Avenue, Sunnyvale, CA
doogkong@gmail.com

Lei Cen
Twitter Inc
1355 Market St #900, San Francisco, CA
lcen@twitter.com

ABSTRACT

To improve mobile application (App for short) user experience, it is very important to inform the users about the apps' privacy risk levels. To address the challenge of incorporating the heterogeneous feature indicators (such as app permissions, user review, developers' description and ads library) into the risk ranking model, we formalize the app risk ranking problem as an exclusive sparse coding optimization problem by taking advantage of features from different modalities via the maximization of the feature consistency and enhancement of feature diversity. We propose an efficient iterative re-weighted method to solve the resultant optimization problem, the convergence of which can be rigorously proved. The extensive experiments demonstrate the consistent performance improvement using the real-world mobile application datasets (totally 13786 apps, 37966 descriptions, 10557681 user reviews and 200 ad libraries).

CCS CONCEPTS

• **Networks** → **Mobile and wireless security**; • **Computing methodologies** → **Supervised learning by classification**.

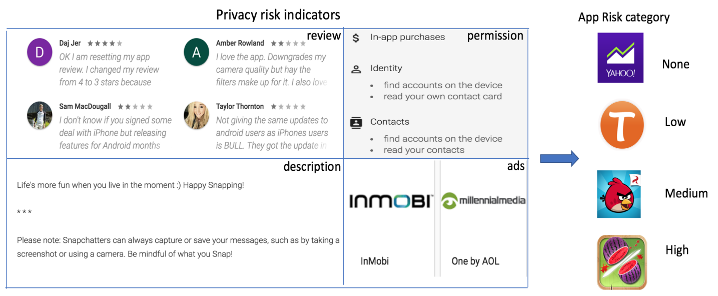
KEYWORDS

App; Mobile; Security; LASSO

ACM Reference Format:
Deguang Kong and Lei Cen. 2019. Mobile App Risk Ranking via Exclusive Sparse Coding. In *Proceedings of the 2019 World Wide Web Conference (WWW'19), May 13–17, 2019, San Francisco, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3308558.3313589

1 INTRODUCTION

Improving user experience is an important task to increase app user engagement and monetize mobile apps. Risk management and assessment has been proposed to improving the user experiences from analysis and diagnosis of potential hazards and risk factors that may cause harm to the mobile app users, such as release of location, user preference, contact and even other personal bio-metric information. A privacy risk score quantifies the risk levels of mobile apps, which allows the users to better perceive the risk levels, which has been shown to have a "significant positive effects" [11] to improving user experience.



Daj Jer
OK I am resetting my app review. I changed my review from 4 to 3 stars because

Amber Rowland
I love the app. Downgrades my camera quality but hay the filters make up for it. I also love

Sam MacDougall
I don't know if you signed some deal with iPhone but releasing features for Android months

Taylor Thornton
Not giving the same updates to android users as iPhones users is BULL. They got the update in

- find accounts on the device
- read your own contact card

- find accounts on the device
- read your contacts

Life's more fun when you live in the moment :) Happy Snapping!

***

Please note: Snapchatters can always capture or save your messages, such as by taking a screenshot or using a camera. Be mindful of what you Snap!

Figure 1: Mobile app risk ranking using heterogeneous privacy risk indicators, including user review, description, permission and ads. The demonstrated four apps, i.e., `Yahoo finance`, `Tango message`, `Angry bird`, and `Fruit Ninja` are categorized as `none`, `low`, `medium` and high, respectively.

To efficiently and effectively perceive the risk of apps is a challenging task, which generally requires the deep understanding of mobile app framework, and expert knowledge of app permission and advertisement libraries. Therefore, machine learning technique is naturally applied to learn the risk scores of apps. Given the heterogeneous risk features indicating different attributions of risks, how to combine them to give a comprehensive assessment by leveraging feature correlations, consistency and diversity, is under-explored but highly desirable. Unfortunately, recent works, including permission usage pattern mining [9] [7], app permission prediction from meta-data [28] [19], malicious app detection [15] [39], API link analysis [24], mobile app recommendation [44] [21], deep model compression on mobile device [20] [18], however, do not essentially solve this problem. To bridge this gap, in this paper, we propose a novel approach to rank app risks using heterogeneous features from multiple sources in a unified optimization framework, which considers both intra-view and inter-view feature information while encouraging their competitions. With our method, one can easily understand which feature play more important role for risk ranking and therefore the result is *interpretable* for illustrating the underlying reasons. Further, once the model is well-trained using the labeled app corpus, the model can be easily deployed and served to automatically label the new apps in the market with high accuracy. To summarize, the key contributions are listed as follows.

• To the best of our knowledge, this paper is the first work that explores heterogeneous privacy indicators for app risk ranking, which, we believe, is very important for improving user experience on mobile platforms.

• We formalize the risk ranking problem as a multi-view feature learning problem via exclusive sparse coding strategy, which can

[*]Thanks for Purdue University and Samsung for the support and resources provided while the authors were doing this work. All authors contribute significantly to the project according to ACM policy.

Table 1: Mobile app risk category

Category	Examples
High	Drag Racing, Fruit Ninja, $\cdots$
Medium	Angry bird, $\cdots$
Low	Temple Run 2, Tango message, $\cdots$
Few/None	Wechat, Yahoo Finance, Instagram, $\cdots$

Table 2: Notations used in the paper

notation	description
$X = [x_1, x_2, \cdots, x_n]$	$x_i^v \in \Re^{p_v}$, v-th view of feature
$Y = [y_1, y_2, \cdots, y_n]$	$y_i \in \Re^K$, risk category for app i
w_k^v	the weight for category k with the v-th view indicator

automatically select the most discriminant features by enforcing both inter-view and intra-view feature competitions.

• We derive a *novel* iteratively re-weighted algorithm to tackle the resultant optimization problem, which can actually handle *any* feature group structures, regardless of coherent or exclusive group feature patterns.

• We crawled real-world datasets (totally 13, 786 apps, 37, 966 descriptions, 10, 557, 681 user reviews and 200 ads libraries), and used them to comprehensively evaluate the effectiveness of our approach. Our method not only consistently outperforms the baseline methods, but also offers valuable insights to understand the risk features attributed to different sources.

Notations Let n be the number of data points, p_v be the dimension for data from view v, k be the number of class in dataset, V be the number of total views. For any vector $\mathbf{x} \in \Re^p$, ℓ_q norm of $\mathbf{w}$ is $\|\mathbf{w}\|_q = (\sum_{i=1}^p \|w_i^q\|)^{\frac{1}{q}}$ for $\forall q \in (0, 2)$.

2 PROBLEM STATEMENT

More formally, assume we have n mobile apps, each mobile app is abstracted as a data point $\mathbf{x}_i$ denoting the privacy indicators extracted from the app. In the paper next, we denote the features extracted from v–th privacy indicator ($1 \le v \le V$) as the v-th view feature. In mobile app context, let $\mathbf{x}_i^v \in \Re^{p_v}$ be v-th view feature (*i.e.*, features extracted from permission, user review, description, ads library, *etc*) of a mobile app i. Consider all the mobile apps, $\mathbf{X}^v = [\mathbf{x}_1^v, \mathbf{x}_2^v, \cdots, \mathbf{x}_n^v]$, where each data column vector is $\mathbf{x}_i^v \in \Re^{p_v \times 1}$. The simple concatenation of all views of apps gives:

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}^1 \\ \mathbf{X}^2 \\ \vdots \\ \mathbf{X}^V \end{bmatrix} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n], \quad \mathbf{x}_i = \begin{pmatrix} \mathbf{x}_i^1 \\ \mathbf{x}_i^2 \\ \vdots \\ \mathbf{x}_i^V \end{pmatrix}.$$

with dimension $p = \sum_v p_v$.

Given the task of risk ranking for mobile apps, the risk of mobile app is categorized into four classes[1]:

`High > Medium > Low > None`,

where the risk levels rank from high to none.

Then each mobile is assigned to a single category. For example, *Yahoo Finance* belongs to *none* risk category. Table 1 gives examples of risk levels for different mobile apps. More formally, let $\mathbf{y}_i$ be risk class label for each $\mathbf{x}_i$ then $\mathbf{y}_i \in \Re^K$ where K is number of categories (K=4 in our case). Written in a matrix format, $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n]$, where $\mathbf{Y} \in \Re^{K \times n}$. The goal of multi-view feature learning task is to learn the projection matrix $\mathbf{W} \in \Re^{p \times K}$ such that the risk levels can be projected into features from different sources, i.e.,

$$\mathbf{W} = [\mathbf{w}_1 \cdots \mathbf{w}_K], \quad \mathbf{w}_k = [\mathbf{w}_k^1, \mathbf{w}_k^2, \cdots, \mathbf{w}_k^V]^T, \tag{1}$$

where $\mathbf{w}_k^v$ has the same size as data $\mathbf{x}_i^v$ from view v. Table 2 summarizes the notations used throughout the paper.

[1]This category is based on expert suggestions from Android developers. `privacygrade: https://www.hcii.cmu.edu/research/privacygrade` from CMU adopts the same category.

Table 3: Privacy risk issues

Privacy risk issue	Behavior Descriptions
Spamming	Advertisement in notification bar, ads via email, ads via SMS, Pop-up ads, Fishing, *etc*
Financial	Paid for In-App-Purchase (IAP), but do not get the item, free to premium, *etc*
Over-claimed Permission	Request too much permissions than users' expectations
Data leakage	Access privacy data without user acknowledgement, *e.g.*, user account, contact, location, *etc*

An illustration In the proposed risk ranking framework, after learning the projection $\mathbf{W}$, we can automatically label the risk category for the new apps following the same feature extraction strategy. For each app, the extracted features $\mathbf{x}_i^v$ are from four perspectives including: (i) permission features with one-hot coding denoting the critical resources (e.g., location, contact) that an app allows to access, (ii) developer description features via tf-idf representation reflecting the developers' awareness [30], (iii) user review features using annotated semantics that provide valuable feedback of users by considering users' expectations (shown in Table 3), (iv) advertisement library features via one-hot coding relating to monetizing purpose [8] with users' personal data.

3 HETEROGENEOUS RISK LEARNING

In this section, we propose a new method for risk ranking using the heterogeneous features. The key idea of our approach to leverage the rich information from different views of features that indicate different aspects of risk of apps while considering both inter-view feature competitions and intra-view feature correlations, which is guided by the following key observations.

(i) For each app risk level, the discriminant features for decision could be different. For example, the dominant features that are used to determine *high* and *none* categories might be "location" and "game", respectively.

(ii) Given the app risk level, some views of features are more important than other views. The competition among the features from different view exists, *i.e.*, *not* all features from each view are needed for risk ranking. For example, in terms of category *low* and *medium*, *advertisement* features are generally more important than other features because mobile apps usually access or even send out users' data stealthily whereas "*permission*" features may be more important for differentiating *high* from *medium* category.

(iii) For the features from the same view, there exist feature correlations and redundancies. From information gain perspective, *less is more.* Thus Feature selection is necessary to select the discriminant features in each view, so as to achieve the minimum redundancy while maintaining maximum relevance at intra-view level.

Based on the above observations, in order to find the projection matrix $\mathbf{W} \in \Re^{p \times K}$ that can map the input features into the risk category, we propose to solve the following objective function:

$$\min_{\mathbf{W}} f(\mathbf{W}; \mathbf{X}, \mathbf{Y}) + \alpha \|\mathbf{W}\|_G + \beta \|\mathbf{W}\|_{EG} \tag{2}$$

where $f(\mathbf{W}; \mathbf{X}, \mathbf{Y})$ is a loss function given multi-view data matrix $\mathbf{X} \in \Re^{p \times n}$ and class labels $\mathbf{Y} \in \Re^{K \times n}$, which is used to minimize the distance between the observations and prediction. Any loss function can be applied here, e.g., cross-entropy loss, logistic loss, hinge loss, etc. For example, with cross-entropy loss, the objective function is:

$$f(\mathbf{W}; \mathbf{X}, \mathbf{Y}) = \sum_{ik} -Y_{ik} log \widehat{Y_{ik}}, \tag{3}$$

$$\widehat{Y_{ik}} = \frac{1}{1 + \exp^{-\mathbf{w}_k^T \phi(\mathbf{x}_i, \theta)}}, \tag{4}$$

where $\phi(\mathbf{x}_i, \theta)$ represents features after projection, either can be in the original feature space, i.e.,

$$\phi(\mathbf{x}_i, \theta) = \mathbf{x}_i, \tag{5}$$

or in projected space after multiple layer non-linear projection:

$$\phi(\mathbf{x}_i, [\theta, \varphi]) = \varphi_L\Big(\theta_L^t[...[\varphi_1(\theta_1^t \mathbf{x}_i)]]\Big), \tag{6}$$

where L is the number of layers, θ_ℓ, φ_ℓ is the feature projection and activation function (such as ReLu, tanh function) for layer ℓ.

The regularization terms are used to incorporate both *local intra-view* via $\|\mathbf{W}\|_{EG}$ and *global inter-view* information via $\|\mathbf{W}\|_G$ given the projection $\mathbf{W} = [\mathbf{w}_1 \mathbf{w}_2 \cdots \mathbf{w}_K]$, i.e.,

$$\|\mathbf{W}\|_G = \sum_{k=1}^{K} \sum_{v=1}^{V} \|\mathbf{w}_k^v\|_2, \quad \|\mathbf{W}\|_{EG} = \sum_{k=1}^{K} \sum_{v=1}^{V} \|\mathbf{w}_k^v\|_1^2. \tag{7}$$

Essentially, we explore both *local intra-view* and *global inter-view* features for multi-view feature selection purpose.

Mathematical Explanation In optimization of $\mathbf{W}$, we optimize each $\mathbf{w}_k (1 \le k \le K)$ separately, because it is corresponding to the feature coefficient computed from class k, independent of $\mathbf{w}_\ell$ from class ℓ, $\forall k \neq \ell$. This indicates that the selected features are unique to each category (as evident in observation i).

(i) $\|\mathbf{W}\|_G$ is used to enforce feature competitions at inter-view level. As a result, features from several views are selected. This term has the same structure as that of group lasso [41] or $\ell_{2,1}$ norm [22] but is applied for multi-view setting. The "group" concept is naturally extended for each view, instead of traditional feature selection ([1, 26]) (as evident in observation ii).

(ii) $\|\mathbf{W}\|_{EG}$ is used to enforce feature competitions at *intra-view* level, because generally features from the same views, could be highly correlated. Thus, as evident in observation (iii), we let features in the same view compete for survival, *i.e.*, only a few number of features in each view could survive. This is achieved by by applying LASSO exclusively on features in each view.

4 OPTIMIZATION ALGORITHM

Fortunately, all terms of Eq.(2) are convex functions, and thus we can obtain a global optimal solution. It is generally felt that joint $\ell_{2,1}$ and $\ell_{1,2}$ terms are much more difficult to solve than the lasso term via shrinkage thresholding. Existing algorithm can formulate them as a quadratic programming problem [27], which can be solved by interior point method or active set method. However, the computational cost is expensive, which limits its use in practice. Recently, a primal-dual algorithm [38] is proposed to solve the similar problem, which casts the non-smooth problem into a min-max problem. However, the algorithm is a gradient descent type method and converges slowly. Moreover, the algorithm is designed for multi-task learning problem, and cannot be applied directly for both group and exclusive lasso problem.

4.1 Proximal Gradient Optimization

Proximal gradient method (a.k.a FISTA method) [25] is widely used to solve $\min_{\mathbf{W}} f(\mathbf{W}) + \Omega(\mathbf{W})$ problem. Here we adopt this method to solve Eq.(2). The key idea of proximal gradient method is to optimize $\mathbf{W}^t$ at each iteration t, *i.e.*,

$$\begin{aligned}\mathbf{W}^{t+1} = \arg\min_{\mathbf{U}} \quad & f(\mathbf{W}^t) + \nabla f(\mathbf{W}^t)^T(\mathbf{U} - \mathbf{W}^t) \\ & + \frac{L}{2}\|\mathbf{U} - \mathbf{W}^t\|_F^2 + \Omega(\mathbf{U}) \\ = \arg\min_{\mathbf{U}} \quad & \frac{1}{2}\|\mathbf{U} - \mathbf{A}\|_F^2 + \frac{1}{L}\Omega(\mathbf{U}),\end{aligned} \tag{8}$$

where $\mathbf{A} = \mathbf{W}^t - \frac{1}{L}\nabla f(\mathbf{W}^t)$, and L is a parameter chosen at each iteration using certain searching strategy, $\Omega(\mathbf{U})$ is the regularization term and $\Omega(\mathbf{U}) = \alpha\|\mathbf{U}\|_G + \beta\|\mathbf{U}\|_{EG}$. Thus the problem is transformed to the minimization problem of Eq.(8).

Their major computation efforts in each iteration is to compute the gradient of $\nabla f(\mathbf{W}^t)$ depending on $f(\mathbf{W}^t)$. For example, for least square loss, it can be computed within $O(pnK)$. For logistic type loss $g(\mathbf{W})$,

$$g(\mathbf{W})' = g(\mathbf{W})(1 - g(\mathbf{W})). \tag{9}$$

With appropriate choose of step size L, the proximal gradient method will achieve ϵ-optimal solution in $O(\frac{1}{\sqrt{\epsilon}})$ iterations. Then the key step is to solve the associated proximal operator:

$$\min_{\mathbf{U}} \frac{1}{2}\|\mathbf{U} - \mathbf{A}\|_F^2 + \frac{1}{L}\Omega(\mathbf{U}). \tag{10}$$

4.2 Iteratively re-weighted method

We present an iteratively re-weighted algorithm to solve Eq.(10). Let $\mathbf{u}_k$ be k-th column of $\mathbf{U}$, the same for $\mathbf{a}_k$. Notice that

$$\begin{aligned}&\frac{1}{2}\|\mathbf{U} - \mathbf{A}\|_F^2 + \frac{1}{L}\Omega(\mathbf{U}) \\ &= \sum_k \Big[\frac{1}{2}\|\mathbf{u}_k - \mathbf{a}_k\|_2^2 + \sum_{v=1}^{V}(\alpha'\|\mathbf{u}_k^v\|_2 + \beta'\|\mathbf{u}_k^v\|_1^2)\Big],\end{aligned} \tag{11}$$

where $\alpha' = \frac{\alpha}{L}, \beta' = \frac{\beta}{L}$. Then we only need to minimize $J_1(\mathbf{u})$ w.r.t to $\mathbf{u} \in \Re^{p \times 1}$, *i.e.*,

$$J_1(\mathbf{u}) = \frac{1}{2}\|\mathbf{u} - \mathbf{a}\|^2 + \sum_{v=1}^{V}(\alpha'\|\mathbf{u}^v\|_2 + \beta'\|\mathbf{u}^v\|_1^2). \tag{12}$$

To the best of our knowledge, this is the *first* work which can tackle both $\ell_{2,1}$ and $\ell_{1,2}$ terms simultaneously. The idea is to find an auxiliary function for Eq.(12) which can be easily solved. Then the updating rules for $\mathbf{u}$ is derived. Finally, we prove the solution is exactly the optimal solution we are seeking for the original problem. Since Eq.(12) is convex, the optimal solution is exact the global optimal solution. The algorithm is named as *iteratively reweighted method*, because it updates **u** iteratively, until the algorithm converges. Moreover, our algorithm can be easily extended to handle arbitrary feature structures, such as linear structure [40], tree structure [23], graph structure [16], forest structure [6], etc

4.2.1 *Algorithm.*

Instead of directly optimizing Eq.(12), we propose to optimize the following objective (the reasons will be seen immediately below), i.e.,

$$J_2(\mathbf{u}) = \frac{1}{2}||\mathbf{u} - \mathbf{a}||^2 + \frac{1}{2}\alpha'\mathbf{u}^T\mathbf{E}\mathbf{u} + \frac{1}{2}\beta'\mathbf{u}^T\mathbf{F}\mathbf{u}, \tag{13}$$

where $\mathbf{E}, \mathbf{F} \in \Re^{p\times p}$ are both diagonal matrices, and given by

$$\mathbf{E}_{ii} = \frac{1}{\sqrt{\sum_{j\in v(i)} \mathbf{u}_j^2}}, \quad \mathbf{F}_{ii} = \frac{2\sum_{j\in v(i)} |\mathbf{u}_j|}{|\mathbf{u}_i|}, \tag{14}$$

where $v(i)$ denotes the view where feature i belongs to. Note the computation of **E**, **F** depends on **u**, thus minimization of **u** depends on both **E**, **F** respectively. In the following, we propose an iteratively re-weighted algorithm to find out the optimal global solution for **u**, where in each iteration, **u** is updated using current **E**, **F**, and **E**, **F** are updated using current **u**. This process is iterated until the algorithm converges. Taking the derivative of Eq.(13) w.r.t **u** and set it to zero. We have

$$\frac{\partial J_2}{\partial \mathbf{u}} = \mathbf{u} - \mathbf{a} + \alpha'\mathbf{E}\mathbf{u} + \beta'\mathbf{F}\mathbf{u} = 0. \tag{15}$$

Thus we obtain the optimal solution for Eq.(13) is given by,

$$\mathbf{u} = (\mathbf{I} + \alpha'\mathbf{E} + \beta'\mathbf{F})^{-1}\mathbf{a}. \tag{16}$$

Note **E**, **F** are all diagonal matrix, and thus the $(\mathbf{I} + \alpha'\mathbf{E} + \beta'\mathbf{F})^{-1}$ can be efficiently computed, where $\mathbf{I} \in \Re^{p\times p}$ is identity matrix. Let $\mathbf{e} \in \Re^p, \mathbf{f} \in \Re^p, e_i = E_{ii}, f_i = F_{ii}, \mathbf{Q} = (\mathbf{I} + \beta_1\mathbf{E} + \beta_2\mathbf{F})^{-1}$, then **Q** is a diagonal matrix and given by,

$$\mathbf{u} = \mathbf{Q}\mathbf{a}, \quad \mathbf{Q}_{ii} = \frac{1}{1 + \alpha' e_i + \beta' f_i}. \tag{17}$$

Using the updating rule of Eq.(17), we can obtain the global optimal solution for Eq.(12). To summarize, Algorithm 1 gives the whole updating process of each individual variable **u**, **E**, **F**.

Algorithm 1 Iterative reweighted algorithm to solve Eq.(12)

```
Input: a, $\alpha'$, $\beta'$
Output: u
Procedure:
1: $t = 0$.
2: while Not converge do
3:     Update $\mathbf{u}^t$ using Eq.(17).
4:     Update $e_i^t = \frac{1}{\sqrt{\sum_{j\in v(i)} u_j^2}}, f_i^t = \frac{2\sum_{j\in v(i)} |\mathbf{u}_j|}{|\mathbf{u}_i|}$ of Eq.(14)
5:     $t = t + 1$.
6: end while
```

Time complexity analysis In practice, we find the above algorithm converges very fast. Typically, it converges to global optimal solution within 10-20 iterations (to precision 1e-6). Moreover, the running time of our algorithm is also very fast, because both **E** and **F** are diagonal matrices. In each iteration, we need to update **u**, **e**, **f** iteratively, which takes time cost $O(p), O(pV), O(pV)$, respectively, where V is the number of view. Thus the total time cost of each iteration is $O(pV)$, which is linear w.r.t the number of features and the number of groups. The convergence analysis is left in Appendix.

Table 4: Descriptions of several mobile datasets crawled in `Google play` at different periods, including user reviews and textual info. The ads and permission are encoded with one-hot coding.

Dataset	#app	#comment	# textual info
D1	6,234	5,028,359	18,157
D2	6,940	4,958,209	16,357
D3	612	571,113	3,452

5 RELATED WORKS

Structure sparsity (e.g.,LASSO [32], group LASSO [41], [37]) has been widely applied for feature learning, which can be extended for processing multi-view feature learning problem. Co-training [2] is widely applied to train two or more separate classifiers by exploring multi-view features. Multiple kernel learning [12] via a combination on the kernel-based method, multi-view deep ranking [4], joint subspace learning by learning a latent view shared by multi-view features [31], multi-Layer multi-View classification for Alzheimer's disease [42], deep heterogeneous feature learning for face recognition [3], multi-view correlated feature learning by uncovering shared component [35] [17], *etc* are also proposed for handling different multi-view learning problems. More detailed survey of multi-view learning can be found in [43], [34]. None of the above works, however, tackles the problem of heterogeneous data ranking problem in the context of app risk ranking. Moreover, most of the above works focus on image data and structured data. Due to the difference between image data and text data (and other unstructured data) investigated in this work, we offer a novel approach for capturing feature correlations and competitions across different views for both text and unstructured data.

6 EXPERIMENT RESULTS

Dataset Collection & Preparation We collected our dataset from *Google Play*. On *Google Play*, user reviews about Apps that he/she used are publicly available. Once we obtain the Google ID of a user, we can locate all Apps the user has reviewed. Therefore, we obtain a list of Google user IDs and write a crawler to retrieve all rated Apps of these users. We collect three datasets at different period in 2017. The first dataset (D1) is collected from Google Play at March which contains 6,234 apps. The second dataset (D2) contains 6,940 apps at July and the third dataset (D3) is collected from top 1000 most popularly apps (by removing the overlaps with D1 and D2). There are no overlaps for the above datasets, and the apps are from a wide range of functionalities, including `game, news, music, social, movies/TV, travel, life,` *etc.* The details are summarized in Table 4.

Multi-view feature pool F For each retrieved App, we crawled its reviews, developer descriptions and apk file as in [33]. Therefore, we have V=4 types of privacy indicators: including permission, user-review, descriptions and advertisement. For each mobile app,

Table 5: Comparisons of our method against other methods in terms of classification performance (mean ± standard deviation). Compared methods: features from each view: Ads library, permission, user comments, description, simple concatenation of all features, and other baselines on D1 dataset.

Method	dataset: D1
Ads library (only)	64.38 ± 1.63
Permission (only)	81.23 ± 2.12
User comments (only)	78.65 ± 2.01
Description (only)	49.23 ± 1.48
Feature concatenation	76.12 ± 2.31
Group [41]	85.23 ± 1.12
LASSO [32]	81.97 ± 2.12
$\ell_{2,1}$ [1]	84.23 ± 1.75
HMNB [29]	85.11 ± 0.34
GBDT [10]	82.47 ± 0.45
MVLR [36]	84.74 ± 0.17
MCCA [14]	83.19 ± 0.45
FCL [35]	85.87 ± 0.31
GElvL (logistic+our method)	88.35 ± 1.68
GElvNL (MLP+our method)	**88.92 ± 0.79**

Table 6: Comparisons of our method against other methods in terms of classification performance (mean ± standard deviation) on D2 dataset.

Method	dataset:D2
Ads library (only)	65.17 ± 1.75
Permission (only)	76.23 ± 2.31
User comments (only)	80.15 ± 1.54
Description (only)	64.56 ± 1.78
Feature concatenation	78.10 ± .93
Group [41]	86.55 ± 1.23
LASSO [32]	83.24 ± 0.78
$\ell_{2,1}$ [1]	85.87 ± 2.12
HMNB [29]	84.23 ± 1.12
GBDT [10]	86.34 ± 0.75
MVLR [36]	86.17 ± 0.29
MCCA [14]	85.23 ± 0.51
FCL [35]	84.29 ± 0.43
GElvL (logistic+our method)	90.47 ± 1.70
GElvNL (MLP+our method)	**91.25 ± 0.92**

the permission feature is encoded using one-hot encoding as 235-dimensional binary vector[2]. We adopt the annotation tool [5] developed to automatic label the user review at app-level with 11 privacy binary labels. The description texts are properly preprocessed and the *tf-idf* feature for each term w in App i's description is computed as $log(TF_w^i + 1)log(\frac{\#App}{DF_w+1})$, where TF_w^i is the term frequency of w for App i's description and DF_w is the number of Apps that have w in their descriptions. Regarding the advertisement feature, we use the method proposed in [13] as a guideline to implement our own analysis tool to extract the ads libraries and then generate the features using one-hot encoding. We carefully generate these features in experiments, and we believe feature construction and feature engineering are important parts in building the model. That is also the reason why we generate the heterogeneous features for app risk categorization.

For the ground-truth of risk category results, we use majority voting[3] based on three experts: (1) two android experts from the major Android phone company; (2) CMU `privacygrad`[4]. If all three reach an agreement, then the label is consistent. If two of them have the same decision, we use the agreement as the final label. If three of them are different, we use the *median* value of their votes as the labeling result.

6.1 Experimental setting

After generating the heterogeneous features, we learn the projection **W** in our model. Parameters of α, β are searched in the range of $(10^{-5}, 10^{-4}, \cdots, 10^{4}, 10^{5})$. We feed the learned **W** for classification purpose. We compare our method against the following methods:

[2]Android system has 235 permissions

[3]We are not saying our decision is perfect, but at least is a good solution to eliminate the bias based on the expert voting.

[4]`https://www.hcii.cmu.edu/research/privacygrade`

• Feature sets from each individual view: ads, user-review, description and permission. We view the result from each single-view as a baseline method.

• A simple concatenation of all features (shown as *CF*) and then feed all the features into logistic regression classifier. Notice that these heterogeneous features have different scales and dimensions, and therefore some feature may dominate the classification process while others are diluted.

• Features learned from only group lasso method (shown as *Group*) (i.e., $\beta = 0$ in Eq.(2) using standard logistic loss, where features in each view are put in the same group, without any distinction.

• Feature learning from standard structural sparsity with logistic loss functio using ℓ_{21}-based feature selection [1].

• Feature learning from standard LASSO (shown as *LASSO*) with logistic loss.

• Mixture of Hierarchical Naive Bayes model[5] [29] with all features feeding into the model, which automatically correlates the risk ranking with the features in a generative way.

• Gradient boost decision tree (GBDT) [10] with an ensemble of a series of decision trees using all the heterogeneous features for categorization purpose. The tree number is set 30–50, number of leaf node=30, learning rate=0.8 and sampling rate=0.8.

• Multi-view unified subspace learning (MVLR) [36] jointly conducts multi-view representation learning and subspace learning.

• Multiview Canonical Correlation Analysis (MCCA) [14] aims to find the common representation that maximizes class separation while minimizing the noises.

• Feature correlation learning (FCL) [35] learns the individual information in each view and captures feature correlations among multiple views after learning the shared component.

[5]However, Peng et al [29] only applied this model to distinguish benign apps from malicious apps only using `permission` features.

Table 7: Comparisons of our method against other methods in terms of classification performance (mean ± standard deviation) on D3 dataset.

Method	dataset:D3
Ads library (only)	60.08 ± 2.31
Permission (only)	75.72 ± 1.75
User comments (only)	74.91 ± 0.81
Description (only)	51.13 ± 0.35
Feature concatenation	70.43 ± 1.50
Group [41]	79.23 ± 0.47
LASSO [32]	78.41 ± 1.56
$\ell_{2,1}$ [1]	79.98 ± 0.32
HMNB [29]	76.74 ± 0.83
GBDT [10]	78.08 ± 1.55
MVLR [36]	75.47 ± 0.32
MCCA [14]	77.98 ± 0.10
FCL [35]	79.05 ± 0.27
GElvL (logistic+our method)	82.32 ± 1.29
GElvNL (MLP+our method)	**82.96 ± 2.44**

• Our method with two versions: (i) GElvL:using logistic loss in Eq.(2); (ii) gElvNL: we use multi-layer perception (MLP) with 3-layers and ReLu activation function in Eq.(2).

6.2 Risk ranking result comparisons

On each dataset, we do five-fold cross validation, i.e., randomly select 80% apps as the labeled apps, and use the rest ones for testing the performance of different algorithms. The above process is repeated for 5 times before reporting the average performances. We use the measurement *accuracy* to evaluate the performance of risk ranking. The performance using 5-fold cross validation are shown in Tables. 5, 6 and 7.

We make several important observations from experiment results. **(i)** Proposed exclusive feature learning method consistently performs better than the other baselines by exploiting the feature correlations while promoting feature diversity. **(ii)** Compared to the performance from the best single-view features, exclusive feature fusion provides better results. **(iii)** For each single-view feature set, their performances vary differently. **(iv)** The performance uplift over group lasso is 4%– 5% on average. **(v)** Our method outperforms the HMNB because simple concatenation will treat features of different views equally, and therefore the learned parameters are biased due to the misalignment among different feature space. GBDT is popularly used for web ranking, however, its performance degrades when it is applied for the classification instead of a typical point-ranking task in the context of mobile apps. **(vi)** Subspace and feature correlation learning generally performs well on image datasets while our method excels on text and unstructured features for mobile apps. **(vii)** Our method gives slightly better performance when applying MLP to embed features in non-linear space, compared to logistic loss.

Discussions on feature types (i) On all datasets, permission indicator contributes the most for ranking the risks of mobile apps, which explains why researches [11] use the permission as the only risk ranking indicator. However, permission indicator is not the only contributor for risk ranking. If multi-modal features are used, the performance is improved. This suggests the necessity of using multi-modal features. **(ii)** Besides permission indicator, the user review contributes the second most for the risk ranking. From users' perspective, the user reviews reveal how the users feel about the app based on their personal experiences. Users' expectation plays a big role in on how much the users can tolerate the apps' behavior, and understand the apps' security risks. **(iii)** Ads-library is also an important indicator to infer the security risks of mobile apps, since the ads libraries communicate with the ads network to get and show ads, based on the app context and users' personal information such as location, context, etc. We believe the ads library will bring serious privacy risk for a mobile app. **(iv)** It is generally felt that developers' description may be more helpful to infer the risks of mobile apps. Our results, however, indicate that description is not as important as we think for evaluating the risks of different mobile apps. This is due to the fact that most of the descriptions are related to the functionality and quality of the app, while only few descriptions mention security/privacy concerns.

Case study. For a game app named `Cut the Rope FULL FREE` [6], it falls into `high` risk category because it loads many ads libraries such as `facebook`, `admob`, `flurry`, *etc*; it requires location, account and other sensitive resources; and users also complain about the ads and finance issues.

7 CONCLUSION

This paper presents a novel approach to learn app risk from heterogeneous features. We address the automatic feature learning problem from features generated from different views, where $\ell_{2,1}$-norm is enforced to select features at inter-view level, and exclusive $\ell_{1,2}$ is enforced to select features at intra-view level. We propose an effective iterative re-weighted algorithm to solve the problem, whose convergence is rigorously proved. We apply the method to solve the mobile app risk ranking problem, and extensive experiments on mobile app risk categorization tasks demonstrate the good performance of our method.

8 APPENDIX: ALGORITHM ANALYSIS

Theorem 8.1. *Under the updating rule of Eq.(16),* $J_1(\mathbf{u}^{t+1}) - J_1(\mathbf{u}^t) \leq 0$.

To prove Theorem 1, we need two lemmas.

Lemma 8.2. *Under the updating rule of Eq.(16),* $J_2(\mathbf{u}^{t+1}) < J_2(\mathbf{u}^t)$.

Lemma 8.3. *Under the updating rule of Eq.(16),*

$$\Big(J_1(\mathbf{u}^{t+1}) - J_1(\mathbf{u}^t)\Big) \leq \Big(J_2(\mathbf{u}^{t+1}) - J_2(\mathbf{u}^t)\Big). \tag{18}$$

REFERENCES

[1] A. Argyriou, T. Evgeniou, and M. Pontil. 2008. Convex multi-task feature learning. *Machine Learning* 73, 3 (2008), 243–272.

[2] A. Blum and T. Mitchell. 1998. Combining labeled and unlabeled data with co-training. *Proc. Comp. Learning Theo. (CLT1998) pp.92-100* (1998).

[3] Navaneeth Bodla, Jingxiao Zheng, and et al. 2017. Deep Heterogeneous Feature Fusion for Template-Based Face Recognition. *CoRR* abs/1702.04471 (2017). http://arxiv.org/abs/1702.04471

[4] Guanqun Cao, Alexandros Iosifidis, Moncef Gabbouj, Vijay Raghavan, and Raju Gottumukkala. 2018. Deep Multi-view Learning to Rank. *CoRR* abs/1801.10402 (2018).

[6] `https://play.google.com/store/apps/details?id=com.zeptolab.ctr.ads&hl=en`

[5] L Cen, L Si, N Li, and H Jin. 2014. User Comment Analysis for Android apps and CSPI Detection with Comment Expansion. In *SIGIR*.

[6] C. Chen, Y. Li, and J. Huang. 2012. Learning with Forest Sparsity. *CoRR* abs/1211.4657 (2012).

[7] Zhenpeng Chen, Xuan Lu, Wei Ai, Huoran Li, Qiaozhu Mei, and Xuanzhe Liu. 2018. Through a Gender Lens: Learning Usage Patterns of Emojis from Large-Scale Android Users. In *Proceedings of the 2018 World Wide Web Conference (WWW '18)*. 763–772.

[8] W. Enck, D. Octeau, P. Mcdaniel, and S. Chaudhuri. 2011. A study of Android application security. In *In Proc. USENIX Security Symposium*.

[9] M. Frank, B. Dong, A. P. Felt, and D. Song. 2012. Mining Permission Request Patterns from Android and Facebook Applications. *CoRR* abs/1210.2429 (2012).

[10] Jerome H. Friedman. 2000. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of Statistics* 29 (2000), 1189–1232.

[11] C. S. Gates, J. Chen, N. Li, and R. W. Proctor. 2014. Effective Risk Communication for Android Apps. *IEEE Trans. Dependable Sec. Comput.* 11, 3 (2014), 252–265.

[12] Mehmet Gönen and Ethem Alpayd. 2011. Multiple Kernel Learning Algorithms. *Journal of Machine Learning Research* 12 (jul 2011), 2211–2268.

[13] M. C. Grace, W. Zhou, X. Jiang, and A.R. Sadeghi. 2012. Unsafe exposure analysis of mobile in-app advertisements. In *WISEC*. 101–112.

[14] David R. Hardoon and etc. 2004. Canonical Correlation Analysis: An Overview with Application to Learning Methods. *Neural Computation* 16 (2004), 2639–2664.

[15] Shifu Hou, Yanfang Ye, Yangqiu Song, and Melih Abdulhayoglu. 2017. HinDroid: An Intelligent Android Malware Detection System Based on Structured Heterogeneous Information Network. In *KDD*. ACM, 1507–1515.

[16] L. Jacob, G. Obozinski, and J.-P. Vert. 2009. Group lasso with overlap and graph lasso. In *ICML*. 55.

[17] Xiao-Yuan Jing, Rui-Min Hu, Yang-Ping Zhu, Shan-Shan Wu, Chao Liang, and Jing-Yu Yang. 2014. Intra-view and Inter-view Supervised Correlation Analysis for Multi-view Feature Learning. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI'14)*. 1882–1889.

[18] Deguang Kong. 2017. Science Driven Innovations Powering Mobile Product: Cloud AI vs. Device AI Solutions on Smart Device. *CoRR* abs/1711.07580 (2017). arXiv:1711.07580 http://arxiv.org/abs/1711.07580

[19] Deguang Kong, Lei Cen, and Hongxia Jin. 2015. AUTOREB: Automatically Understanding the Review-to-Behavior Fidelity in Android Applications. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security (CCS '15)*. ACM, New York, NY, USA, 530–541. https://doi.org/10.1145/2810103.2813689

[20] Dawei Li, Xiaolong Wang, and Deguang Kong. 2018. DeepRebirth: Accelerating Deep Neural Network Execution on Mobile Devices. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018.* 2322–2330.

[21] Bin Liu, Deguang Kong, Lei Cen, Neil Zhenqiang Gong, Hongxia Jin, and Hui Xiong. 2015. Personalized Mobile App Recommendation: Reconciling App Functionality and User Privacy Preference. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining (WSDM '15)*. ACM, New York, NY, USA, 315–324. https://doi.org/10.1145/2684822.2685322

[22] J. Liu, S. Ji, and J. Ye. 2012. Multi-Task Feature Learning Via Efficient l2,1-Norm Minimization. *CoRR* abs/1205.2631 (2012).

[23] J. Liu and J. Ye. 2010. Moreau-Yosida Regularization for Grouped Tree Structure Learning. In *NIPS*. 1459–1467.

[24] Yun Ma, Ziniu Hu, Yunxin Liu, Tao Xie, and Xuanzhe Liu. 2018. Aladdin: Automating Release of Deep-Link APIs on Android. In *Proceedings of the 2018 World Wide Web Conference (WWW '18)*. 1469–1478.

[25] Y. Nesterov. 2007. Gradient methods for minimizing composite objective function. *ECORE Discussion Paper* (2007).

[26] F. Nie, H. Huang, X. Cai, and C.H.Q. Ding. 2010. Efficient and Robust Feature Selection via Joint $\ell_{2,1}$-Norms Minimization. In *NIPS*. 1813–1821.

[27] S.J Nocedal, J.; Wright. 2006. *Numerical Optimization.* Springer-Verlag, Berlin, New York.

[28] R. Pandita, X. Xiao, W. Yang, and et al. 2013. WHYPER: Towards Automating Risk Assessment of Mobile Applications. In *USENIX Security*. 527–542.

[29] Hao Peng, Chris Gates, Bhaskar Sarma, and et al. 2012. Using Probabilistic Generative Models for Ranking Risks of Android Apps. In *CCS*. 241–252.

[30] Z. Qu, V. Rastogi, X. Zhang, and et al. 2014. AutoCog: Measuring the Description-to-permission Fidelity in Android Applications. In *ACM CCS*. 1354–1365.

[31] Abhishek Sharma. 2012. Generalized Multiview Analysis: A Discriminative Latent Space. In *CVPR*. IEEE Computer Society, Washington, DC, USA, 2160–2167. http://dl.acm.org/citation.cfm?id=2354409.2355114

[32] R. Tibshirani. 1996. Regression shrinkage and selection via the LASSO. *J. Royal. Statist. Soc B.* 58 (1996), 267–288.

[33] Nicolas Viennot, Edward Garcia, and Jason Nieh. 2014. A Measurement Study of Google Play. In *ACM SIGMETRICS*. ACM, 221–233. https://doi.org/10.1145/2591971.2592003

[34] Chang Xu, Dacheng Tao, and Chao Xu. 2013. A Survey on Multi-view Learning. *CoRR* abs/1304.5634 (2013).

[35] Xiaowei Xue, Feiping Nie, Sen Wang, Xiaojun Chang, Bela Stantic, and Min Yao. 2017. Multi-View Correlated Feature Learning by Uncovering Shared Component. In *AAAI*. 2810–2816.

[36] Zhe Xue, Guorong Li, and Qingming Huang. 2016. Joint Multi-View Representation Learning and Image Tagging. In *AAAI*. 1366–1372.

[37] Haichuan Yang, Yijun Huang, Lam Tran, Ji Liu, and Shuai Huang. 2016. On Benefits of Selection Diversity via Bilevel Exclusive Sparsity. In *CVPR*. 5945–5954.

[38] T. Yang, R. Jin, M. Mahdavi, and S. Zhu. 2012. An Efficient Primal-Dual Prox Method for Non-Smooth Optimization. *CoRR* abs/1201.5283 (2012).

[39] Wei Yang, Deguang Kong, Tao Xie, and Carl A. Gunter. 2017. Malware Detection in Adversarial Settings: Exploiting Feature Evolutions and Confusions in Android Apps. In *Proceedings of the 33rd Annual Computer Security Applications Conference (ACSAC 2017)*. ACM, New York, NY, USA, 288–302. https://doi.org/10.1145/3134600.3134642

[40] L. Yuan, J. Liu, and J. Ye. 2011. Efficient Methods for Overlapping Group Lasso. In *NIPS*. 352–360.

[41] M. Yuan and M. Yuan. 2006. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B* 68 (2006), 49–67.

[42] Changqing Zhang, Ehsan Adeli, Tao Zhou, Xiaobo Chen, and Dinggang Shen. 2018. Multi-Layer Multi-View Classification for Alzheimer's Disease Diagnosis. In *AAAI*.

[43] Jing Zhao, Xijiong Xie, Xin Xu, and Shiliang Sun. 2017. Multi-view Learning Overview. *Inf. Fusion* 38 (2017), 43–54.

[44] H. Zhu, H. Xiong, Y. Ge, and E. Chen. 2014. Mobile App Recommendation with Security and Privacy Awareness. In *KDD*. ACM, 951–960.