Representer Point Selection for Explaining Regularized High-dimensional Models

Che-Ping Tsai [1] **Jiong Zhang** [2] **Hsiang-Fu Yu** [2] **Eli Chien** [3] **Cho-Jui Hsieh** [2 4] **Pradeep Ravikumar** [1]

Abstract

We introduce a novel class of sample-based explanations we term *high-dimensional representers*, that can be used to explain the predictions of a regularized high-dimensional model in terms of importance weights for each of the training samples. Our workhorse is a novel representer theorem for general regularized high-dimensional models, which decomposes the model prediction in terms of contributions from each of the training samples: with positive (negative) values corresponding to positive (negative) impact training samples to the model's prediction. We derive consequences for the canonical instances of ℓ_1 regularized sparse models, and nuclear norm regularized low-rank models. As a case study, we further investigate the application of low-rank models in the context of collaborative filtering, where we instantiate high-dimensional representers for specific popular classes of models. Finally, we study the empirical performance of our proposed methods on three real-world binary classification datasets and two recommender system datasets. We also showcase the utility of high-dimensional representers in explaining model recommendations.

1. Introduction

Sample-based explanations aim to explain a machine learning model's prediction by identifying the most influential training samples that led to the prediction. This is usually done by measuring the influence of each training sample on the model's prediction scores. The explanations not only assist users in understanding the rationale behind the prediction, but also allow model designers to debug or de-bias the training data (Kong et al., 2021; Thimonier et al., 2022).

To measure the impact of each training sample on the prediction score, a classical technique is to compute the derivative of the prediction score with respect to each training instance using implicit function theory, an approach also known as influence functions (Cook & Weisberg, 1980; Koh & Liang, 2017). However, computing the influence function requires the inversion of the Hessian matrix, causing significant scalability issues when handling large models. To compute sample-based explanations in an efficient manner, another method called **Representer Point Selection** has been developed (Yeh et al., 2018). This method is based on the classical representer theorem (Schölkopf et al., 2001), which states that a regularized empirical risk minimizer over a reproducing kernel Hilbert space (RKHS) can be decomposed into a linear combination of kernel functions evaluated on each training sample. While functions parameterized with neural networks do not necessarily lie in a pre-specified RKHS, Yeh et al. (2018) propose to treat the last layer of a neural network as a linear machine and the remaining part as a fixed feature encoder. Upon fine-tuning the last layer with ℓ_2 regularization, the representer theorem can then be applied, allowing us to obtain importance scores of the training data. In this development, the use of ℓ_2 regularization served as a RKHS norm with respect to linear kernels, which was key to recruiting the representer theorem.

However, ℓ_2 regularizers are not always suitable for high-dimensional models where the number of parameters might even be larger than the number of samples, and where the model parameters might lie in a lower dimensional subspace. In such settings, in order for the resulting estimators to have strong statistical guarantees, it is often critical to employ high-dimensional regularizations that encourage the model parameter to lie in such lower-dimensional structured subspaces (Negahban & Wainwright, 2011). Two canonical instances of such high-dimensional regularizers include the ℓ_1 norm regularization that encourages parameter vectors to have sparse structure, and the nuclear norm regularization imposes low-rank structure on parameter matrices. The caveat however is that these regularizations cannot typically be cast as RKHS norms, and thus the classical representer theorem does not apply. Therefore, it remains unclear how to select representer points for high-dimensional models, despite the widespread use of high-dimensional models in practical applications such as compressed sensing (Donoho,

[1]Carnegie Mellon University [2]Amazon, USA [3]University of Illinois Urbana-Champaign [4]University of California, Los Angeles. Correspondence to: Che-Ping Tsai <chepingt@andrew.cmu.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

2006) and recommender systems (Candes & Recht, 2008; Recht, 2011).

We first present a general theorem that provides a representer theorem for regularized high-dimensional models, where we leverage the rich structure of the regularization sub-differentials, as well as the analytical framework of Negahban et al. (2012) that associates the regularization functions with a collection of structured low-dimensional subspaces. We term the resulting sample-based explanations for these high-dimensional models as *high-dimensional representers*. As with the original representer points for ℓ_2 regularized models, there is a global importance score per training sample, as well as a local importance score that measures the similarity between the test point and the training sample. But unlike the ℓ_2 regularized case, the representer theorem entails that this local similarity is measured after an appropriate linear projection of the test input and the training sample to the structured model parameter subspace. Thus, even in cases where the model parameters might be quite high-dimensional, the local similarity is quite meaningful, as well as scalable and efficient since it is computed over a much lower dimensional structured subspace.

Given the general theorem, we then derive its consequences for the important settings of sparse vectors with ℓ_1 regularization, and low-rank matrices with nuclear norm regularization, leading to sample-based explanation methods under those high-dimensional regularizers. Equipped with the results, we explore the use of our technique in the context of collaborative filtering, including various specific model instances such as collaborative matrix factorization models (Koren et al., 2009). We also investigate deep neural network variations of these models, the two-tower models (Mao et al., 2021; Li et al., 2019), by treating the final interaction layer is treated as a bilinear matrix factorization model and the other layers are fixed encoders when applying our method. This cannot be done with the ℓ_2 representer methods as the final layer is a product of two matrices. Lastly, we evaluate the empirical performance of the high-dimensional representers on three real-world binary classification datasets and two recommender system datasets. We also demonstrate the practical utility of high-dimensional representers in explaining the recommendations generated by our models.

2. Related Work

Prominent approaches for estimating training data influence to a test point include influence functions (Koh & Liang, 2017), representer point selection (Yeh et al., 2018), and TracIn (Pruthi et al., 2020). Influence functions (Wojnowicz et al., 2016; Khanna et al., 2019; Bae et al., 2022) estimate training sample importance by measuring "how the model's prediction change if we remove a particular training sample and retrain the model." However, computing influence functions requires calculating the inverse of the Hessian matrix. Exact estimation requires time complexity at least quadratic to the number of parameters and is thus unsuitable for large or high-dimensional models (Guo et al., 2020; Hammoudeh & Lowd, 2022; Schioppa et al., 2022).

TracIn quantifies training data importance by measuring similarities between gradient at training and test samples over trajectories (Yeh et al., 2022; Chen et al., 2021). However, their approach only applies to models trained with stochastic gradient descent, which may not be an efficient way for high-dimensional model training. Also, TracIn requires storing and accessing checkpoints of models during training and is not applicable to off-the-shelf models. The most relevant work to ours is the (ℓ_2) representer point selection: Brophy et al. (2022) extends it to explain decision trees using supervised tree kernels. Sui et al. (2021) improves it with local Jacobian expansion. Another line of sample-based explanations relies on repeated retraining (Ghorbani & Zou, 2019; Jia et al., 2019; Kwon & Zou, 2021; Feldman & Zhang, 2020), which are more costly compared to the methods mentioned above since it requires retraining models multiple times.

On the other hand, representer theorems (Schölkopf et al., 2001) in machine learning have targeted non-parametric regression in RKHS. Bohn et al. (2019) connect representer theorems and composition of kernels. Unser (2019) derive general representer theorems for deep neural networks and make a connection with deep spline estimation. Unser et al. (2016) also propose representer theorems for ℓ_1 regularization, but their theorems have a different formulation for a difference purpose: they attribute model parameters to basis on the nonzero coordinates to show that the minimizer is sparse. In our work, we consider a simpler task of explaining regularized high-dimensional models and develop novel representer theorems for this purpose.

3. Preliminary

Before providing our general framework for high-dimensional representers, it is instructive to recall classical machinery in high-dimensional estimation. As Negahban et al. (2012) show, we can think of structure in high-dimensional models as being specified by collections of lower-dimensional subspaces.

Example: Sparse Vectors: Consider the set of s-sparse vectors in p dimensions. For any particular subset $S \subseteq \{1, \ldots, p\}$, with cardinality s, define the subspace: $A(S) = \{\theta \in \mathbb{R}^p \,:\, \theta_j = 0, \quad \forall j \notin S\}$. It can then be seen an s-sparse vector lies in one of the collection of low-dimensional subspaces $\{A(S)\}_{S \subseteq [p]}$.

Example: Low-Rank Matrices: For any matrix $\Theta \in$

$\mathbb{R}^{d_1 \times d_2}$, let $\text{col}(\Theta) \in \mathbb{R}^{d_1}$ its column space, and $\text{row}(\Theta) \in \mathbb{R}^{d_2}$ denote its row space. For a given pair (U, V) or k-dimensional subspaces $U \subseteq \mathbb{R}^{d_1}$ and $V \subseteq \mathbb{R}^{d_2}$, we can define the subspaces: $A(U, V) = \{\Theta \in \mathbb{R}^{d_1 \times d_2} : \text{col}(\Theta) \subseteq U, \text{row}(\Theta) \subseteq V\}$. It can then be seen that any low-rank matrix $\Theta \in \mathbb{R}^{d_1 \times d_2}$ of rank $k \leq \min(d_1, d_2)$ lies in a collection of the low-dimensional subspaces above.

A critical question in such high-dimensional settings is how to automatically extract and leverage such low-dimensional subspace structure. Negahban et al. (2012) showed that so long as regularization functions $r(\cdot)$ satisfy a property known as decomposability with respect to one of the collections of subspaces, regularized empirical loss minimizers yield solutions that lie in a low-dimensional subspace within that collection. Towards defining this, they require another ingredient which is a collection of orthogonal subspaces of parameters with orthogonal structure. For sparse vectors, the orthogonal subspace $B(S) = A(S)^{\perp}$. For low-rank matrices, the orthogonal subspace is $B(U, V) = \{\Theta \in \mathbb{R}^{d_1 \times d_2} : \text{row}(\Theta) \subseteq U^{\perp}, \text{col}(\Theta) \subseteq V^{\perp}\}$. It can be seen that in this case, we have that $B(U, V) \subseteq A^{\perp}(U, V)$, since we do not simply want all orthogonal parameters to the structured subspace, but want orthogonal parameters which are also structured with respect to the collection of subspaces. A regularization $r(\cdot)$ is said to be decomposable with respect to collection of subspaces if for any such structured subspace pair (A, B), we have that: $r(u + v) = r(u) + r(v)\ \forall u \in A, v \in B$. For the case of sparse vector subspaces, the ℓ_1 norm $r(\theta) = \|\theta\|_1$, and for the case of low-rank matrices, the nuclear norm $r(\Theta) = \|\Theta\|_*$ can be shown to be decomposable (Negahban et al., 2012).

The sub-differential of the regularization function can be written as: $\partial r(\theta) = \{u \,|\, r(\theta') - r(\theta) \geq \langle u, \theta' - \theta \rangle, \forall \theta' \in \Theta\}$. In the case of structured parameters above, the sub-differential in turn has additional structure. Suppose (A, B) is the subspace pair corresponding to the structured parameter θ. Then, for any $g \in \partial r(\theta)$, we have that $g = u_\theta + v$, where $u_\theta \in A$ has a unique representation that depends on θ, and $v \in B$. Moreover, there exists a (non-unique) inverse transform $(\partial_\theta r)^+$ of the partial differential, so that $(\partial_\theta r)^+(g) = \theta$, for all $g \in \partial r(\theta)$, with the property that $(\partial_\theta r)^+$ is a positive-definite linear operator, with range within the structured subspace A.

4. Representer Theorem for High-Dimensional Models

We are interested in regularized empirical risk minimizers. Given n training samples $(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_n, y_n) \in \mathcal{X} \times \mathbb{R}$, a loss function $\ell(\cdot, \cdot) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, and parameters of a linear model $\theta \in \boldsymbol{\Theta}$, where $\boldsymbol{\Theta} \subseteq \mathcal{X}$ we consider the following optimization problem:

$$\hat{\theta} = \underset{\theta \in \boldsymbol{\Theta}}{\text{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle x_i, \theta \rangle) + \lambda r(\theta). \tag{1}$$

In the sequel, we assume that the regularization function $r(\cdot)$ is decomposable with respect to some collection of low-dimensional structured subspace, as briefly reviewed in Section 3. Its role is to encourage the model parameter θ to have the appropriate low-dimensional structure, while the hyper-parameter λ balances loss and regularization.

Theorem 1. *(high-dim representer theorem) The minimizer $\hat{\theta}$ of Eqn.(1) can be written as*

$$\hat{\theta} = \sum_{i=1}^{n} \left(-\frac{1}{n\lambda} \ell'(y_i, \langle x_i, \hat{\theta} \rangle) \right) \left((\partial_{\hat{\theta}} r)^+ x_i \right), \tag{2}$$

where $\ell' = \partial \ell / \partial(\langle x_i, \hat{\theta} \rangle)$ denotes the partial derivative of ℓ with respect to its second input variable, and $(\partial_{\hat{\theta}} r)^+$ is the (non-unique) inverse transform of the regularization sub-differential. For any given test sample $x' \in \mathcal{X}$, its prediction can be decomposed according to training samples:

$$\langle x', \hat{\theta} \rangle = \sum_{i=1}^{n} \underbrace{-\frac{1}{n\lambda} \ell'(y_i, \langle x_i, \hat{\theta} \rangle))}_{\textit{global importance}} \underbrace{\langle (\partial_{\hat{\theta}} r)^{\frac{+}{2}} x_i, (\partial_{\hat{\theta}} r)^{\frac{+}{2}} x' \rangle}_{\textit{local importance}}, \tag{3}$$

where $(\partial_{\hat{\theta}} r)^{\frac{+}{2}}$ is the square-root of the sub-differential inverse transform.

Eqn.(3) provides the attribution of each training sample x_i to a test sample x', which can be decomposed into the global importance and local importance. The global importance is a measure of how sensitive the training sample x_i is to the objective and depends on the derivative of the loss function. The local importance measures the similarity between the training sample x_i and the test sample x'.

The local importance similarity focuses on the projection of the data points onto a structured low-dimensional subspace A since the range of the sub-differential inverse transform $(\partial_{\hat{\theta}} r)^{\frac{+}{2}}$ is the structured subspace within which the parameter lies. We can thus think of such high-dimensional model estimation as specifying the local kernel $k(x, x') = \langle (\partial_{\hat{\theta}} r)^{\frac{+}{2}} x, (\partial_{\hat{\theta}} r)^{\frac{+}{2}} x' \rangle$. To see a crucial difference with ℓ_2 regularized models (Yeh et al., 2018), where the local importance is simply an inner product between x_i and x', high-dimensional representers ignore the features in the orthogonal space B since they have no impact on test predictions.

The theorem is derived from solving first-order optimality condition on the low-dimensional subspace A, i.e. one subgradient of the minimizer with respect to the objective equals zero. Next, we utilize the fact that the sub-differential

$\partial r(\hat{\theta})$ has a unique representation in the model subspace A. It allows us to develop the inverse transform operator $(\partial_{\hat{\theta}} r)^+$ and use it to recover the model parameter.

In cases where the inverse transform is non-unique, we would obtain multiple local importance, one for each inverse transform, and we can then take an average of these when computing the local importance.

While the above development was quite abstract, in the following sections, we derive its consequences for the important settings of sparse vectors with ℓ_1 regularization, and low-rank matrices with nuclear norm regularization.

4.1. ℓ_1-regularized Linear Optimization

Based on the general theorem, we derive the representer point selection method for ℓ_1 regularization. We consider the following special case of Eqn.(1):

$$\hat{\theta} = \underset{\theta \in \mathbb{R}^p}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle x_i, \theta \rangle) + \lambda \|\theta\|_1, \tag{4}$$

where the ℓ_1 regularization encourages the model to be sparse. Some examples of Eqn.(4) include ℓ_1-regularized generalized linear models (Tibshirani, 1996), compressed sensing (Donoho, 2006), and sparse estimation of Gaussian graphical models (Yuan & Lin, 2007; Friedman et al., 2008).

We develop the representer theorem for ℓ_1 regularized problems using Theorem 1. In this case, the structural model subspace is specified by the sparse model parameter $\hat{\theta}$, $A(S(\hat{\theta})) = \{\theta \in \mathbb{R}^p : \theta_j = 0, \forall j \notin S\}$, where $S(\hat{\theta})$ denotes a set of coordinates that $\hat{\theta}$ has non-zero values. The orthogonal subspace $B(S(\hat{\theta}))$ is in turn a set of vectors in $\mathbb{R}^p$ whose coordinates on $S(\hat{\theta})$ are zero.

Next, the sub-differential of the ℓ_1 norm is $\partial \|\hat{\theta}\|_1 = \{g \in \mathbb{R}^p | g_i = \text{sign}(\hat{\theta}) \text{ if } \hat{\theta}_i \neq 0, \text{ and } |g_i| \leq 1 \text{ if } \hat{\theta}_i = 0\}$, which has a unique representation, $\text{sign}(\hat{\theta})$, in $A(S(\hat{\theta}))$. Next, the inverse transform can be developed by reconstructing $\hat{\theta}$ in the model subspace and zeroing out the sub-differential in the orthogonal space, where the model parameters are zero. Specifically, we use $(\partial_\theta r)^+(x) = |\hat{\theta}| \odot x, \forall x \in \mathbb{R}^p$, where $|\cdot|$ denotes a coordinate-wise absolute value operator, and $\odot$ denotes element-wise multiplication. Clearly, we have $(\partial_\theta r)^+(g) = \hat{\theta}$ for all $g \in \partial \|\hat{\theta}\|_1$ since $\hat{\theta}_j g_j = \hat{\theta}_j$ if $j \in S(\hat{\theta})$ and $\hat{\theta}_j g_j = 0$ if $j \notin S(\hat{\theta})$. By plugging these notations to Theorem 1, we obtain the following representer theorem for ℓ_1 regularized linear optimization problems.

Corollary 2. *(high-dim representer theorem for ℓ_1-regularizaion) The minimizer $\hat{\theta}$ of Eqn.(4) can be written as*

$$\hat{\theta} = \sum_{i=1}^{n} \left(-\frac{1}{n\lambda} \ell'(y_i, \langle x_i, \hat{\theta} \rangle) \right) \left(|\hat{\theta}| \odot x_i \right), \tag{5}$$

For any given test sample $x' \in \mathbb{R}^p$, its prediction can be decomposed according to training samples:

$$\langle x', \hat{\theta} \rangle = \sum_{i=1}^{n} \underbrace{-\frac{1}{n\lambda} \ell'(y_i, \langle x_i, \hat{\theta} \rangle)}_{\textit{global importance } \alpha_i} \underbrace{\langle \sqrt{|\hat{\theta}|} \odot x_i, \sqrt{|\hat{\theta}|} \odot x' \rangle}_{\textit{local importance}}, \tag{6}$$

where $\sqrt{\cdot}$ is a coordinate-wise square root operation.

With Corollary 2, we can quantify training data influence on a specific test sample (x', y'). The sign of $\alpha_i \langle \sqrt{|\hat{\theta}|} \odot x_i, \sqrt{|\hat{\theta}|} \odot x' \rangle$ indicates whether a training sample (x_i, y_i) has positive or negative influence on the test sample. Also, if a training sample (x_i, y_i) has a large importance value to a test sample x', two conditions must be satisfied: (1) global importance α_i is large (2) $\sqrt{|\hat{\theta}|} \odot x_i$ is close to $\sqrt{|\hat{\theta}|} \odot x'$. That is, x_i and x' are close on the coordinates where the model parameters $\hat{\theta}$ have non-zero values.

4.2. Nuclear-norm Regularized Linear Optimization

We consider the following canonical nuclear norm regularized linear optimization problem with inputs and model parameters being matrices. Given n training samples $(X_1, y_1), \cdots, (X_n, y_n) \in \mathbb{R}^{d_1 \times d_2} \times \mathbb{R}$, a loss function $\ell(\cdot, \cdot) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, and parameters of a linear model $\Theta \in \mathbb{R}^{d_1 \times d_2}$, we consider the following problem:

$$\hat{\Theta} = \underset{\Theta \in \mathbb{R}^{d_1 \times d_2}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(y_i, \langle X_i, \Theta \rangle_F) + \lambda \|\Theta\|_*, \tag{7}$$

where $\langle \cdot, \cdot \rangle_F$ is a Frobenius inner product operator, and $\|\cdot\|_*$ is the Nuclear norm, defined as the sum of ℓ_1 norm of singular values. This formulation has been applied in matrix completion (Candès & Tao, 2010), matrix regression (Yang et al., 2016), and matrix compressed sensing (Eldar & Kutyniok, 2012) with low-rank constraints.

As in Negahban et al. (2012), the low-rank model subspace $A(U, V)$ is specified by a full singular value decomposition (SVD) of the model parameter $\hat{\Theta} = U \Sigma V^\top$, where the columns of $U \in \mathbb{R}^{d_1 \times k}$ and $V \in \mathbb{R}^{d_2 \times k}$ are orthogonal, $\Sigma \in \mathbb{R}^{k \times k}$ is a diagonal matrix, and $k = \text{rank}(\hat{\Theta})$. The orthogonal subspace is $B(U, V) = \{\Theta \in \mathbb{R}^{d_1 \times d_2} : \text{row}(\Theta) \subseteq U^\perp, \text{col}(\Theta) \subseteq V^\perp\}$.

The sub-differential of the nuclear norm (Watson, 1992) is $\partial \|\hat{\Theta}\|_* = \{UV^\top + W : W \in \mathbb{R}^{d_1 \times d_2}, \|W\|_2 \leq 1, WV = \mathbf{0}, U^\top W = \mathbf{0}\}$, which can be decomposed as a unique representation in the model subspace ($UV^\top \in A(U, V)$) and $W \in B(U, V)$ in the orthogonal space. In this case, the inverse transform of sub-differential is not unique: it can be either $(\partial_{\hat{\theta}} r)^+(X) = U \Sigma U^\top X$ or $(\partial_{\hat{\theta}} r)^+(X) = X V \Sigma V^\top$ for any $X \in \mathbb{R}^{d_1 \times d_2}$. One can easily verify that the inverse transform recovers $\hat{\Theta}$, $(\partial_{\hat{\theta}} r)^+(\partial \|\hat{\Theta}\|_*) = \hat{\Theta}$ using the fact that $U^\top U = V V^\top = I_k$. By instantiating the inverse transform to Theorem 1, we obtain the following corollary.

Corollary 3. *(high-dim representer theorem for nuclear-norm regularizaion) Let $U\Sigma V^\top = \hat{\Theta}$ be a full SVD of the minimizer $\hat{\Theta}$ of Eqn.(7). The minimizer of Eqn.(7) can be written as*

$$\hat{\Theta} = \sum_{i=1}^{n} -\frac{1}{n\lambda}\ell'(y_i, \langle X_i, \hat{\Theta}\rangle_F)\left(U\Sigma U^\top X_i\right) \quad (8)$$

$$= \sum_{i=1}^{n} -\frac{1}{n\lambda}\ell'(y_i, \langle X_i, \hat{\Theta}\rangle_F)\left(X_i V^\top \Sigma V\right). \quad (9)$$

For any given test sample $X' \in \mathbb{R}^{d_1\times d_2}$, its prediction can be decomposed according to training samples:

$$\langle X', \hat{\Theta}\rangle = \sum_{i=1}^{n} -\frac{1}{n\lambda}\ell'(y_i, \langle X_i, \hat{\Theta}\rangle))\langle \sqrt{\Sigma}U^\top X_i, \sqrt{\Sigma}U^\top X'\rangle_F \quad (10)$$

$$= \sum_{i=1}^{n} -\frac{1}{n\lambda}\ell'(y_i, \langle X_i, \hat{\Theta}\rangle))\langle X_i V\sqrt{\Sigma}, X' V\sqrt{\Sigma}\rangle_F, \quad (11)$$

where $\sqrt{\Sigma} = diag[\sqrt{\Sigma_{11}}, \cdots, \sqrt{\Sigma_{kk}}]$.

Again, the first term in Eqn.(10) and Eqn.(11), $-\frac{1}{n\lambda}\ell'(y_i, \langle X_i, \hat{\Theta}\rangle))$, is the *global importance* and the second inner product terms are the *local importance*. We first project input matrices X_i and X' onto the column or row spaces by multiplying with $\sqrt{\Sigma}U^\top$ or $V\sqrt{\Sigma}$, respectively, and then computing the Frobenius inner product. This term measures local similarities between a test sample and training samples in the column or row spaces of the minimizer $\hat{\Theta}$.

Unlike Corollay 2, Eqn.(10) and Eqn.(11) provide two distinct ways to decompose the learned model, leading to two different ways for data attribution. We refer to Eqn.(10) as *column-based attribution* and Eqn.(11) as *row-based attribution*, since they compute local importance on the column/row spaces of $\hat{\Theta}$, respectively. The interpretation of these two attributions may depend on applications. For example, as we will show in Corollary 4, the two attributions correspond to user-based attribution and item-based attributions when U and V are user and item embeddings in recommender systems. In other cases, we may take the average of the two local importance.

4.3. Computation of High-dimensional Representers

In this section, we introduce the computation of high-dimensional representers. To explain a model's prediction on x', one needs to compute the high-dimensional representers for the test sample x' with respect to all training samples $\{(x_i, y_i)\}_{i=1}^n$. In practice, we could pre-process the training to accelerate the computation. Recall that high-dimensional representers in Eqn.(3) consist of two components: a global importance $\alpha_i = -\frac{1}{n\lambda}\ell'(y_i, \langle x_i, \hat{\theta}\rangle))$, and a local importance $\langle(\partial_{\hat{\theta}} r)^{\frac{+}{2}} x_i, (\partial_{\hat{\theta}} r)^{\frac{+}{2}} x'\rangle$. At the pre-processing step, we compute the global importances α for all training samples and their projections onto the low-dimensional model space, i.e. $(\partial_{\hat{\theta}} r)^{\frac{+}{2}} x_i$ for all $i \in [n]$.

Note that global importances can be obtained by inferring all training data and calculating their derivatives. The projection operator can usually be obtained from the training stage since the model parameter $\hat{\theta}$ is available in the ℓ_1 case, and the full SVD can usually be obtained from the training stage (Mazumder et al., 2010) in the nuclear norm case. The pre-processing step requires $O(np)$ and $O(nkd_1d_2)$ time for the ℓ_1-norm and nuclear norm cases respectively. We note that in the nuclear-norm case, the pre-processing step typically takes no longer than training the regularized models with a single epoch. This is because the training samples typically need to be projected to the low-dimensional space to calculate the update formula (Hsieh & Olsen, 2014).

Next, to explain a test prediction, we need to (1) project the test sample to the model subspace and (2) compute the inner product between the test and training samples in the model subspace. While step (1) only needs to tackle one sample, step (2) takes $O(np)$ and $O(n\max(d_1, d_2)k)$ time for the ℓ_1-norm and nuclear norm cases respectively.

In many applications of sample-based explanations, such as generating human-understandable explanations, we only care about the top influential samples for a test prediction. This can be significantly sped up by approximate nearest neighbor search algorithms which can be run in sublinear time since we only need to find training samples with the highest inner product values.

5. Applications to Collaborative Filtering (CF)

With the widespread deployment of recommender systems across various online platforms, the significance of explainable recommender systems has grown substantially (Zhang et al., 2020). Studies have indicated that users prefer recommendations that are explainable, and explanation tools are vital for debugging recommendation models (Tintarev & Masthoff, 2007). In this section, we showcase how high-dimensional representers can effectively explain collaborative filtering models and (deep) recommender systems.

Notations: Given a set of users $\mathcal{U}$, a set of items $\mathcal{I}$ and a set of user-item interactions $\mathcal{D} = \{(i,j)| \mid i \in \mathcal{U}, j \in \mathcal{I}, y_{ij}$ is observed $\}$, CF aims to learn a k-dimensional embedding for each user and item, and utilizes inner products between user and item embeddings to predict unknown elements in the matrix.

5.1. Matrix Factorization (MF) with Nuclear Norm Regularization

Matrix factorization with nuclear norm regularizations (Candès & Tao, 2010; Candes & Recht, 2012) is a success-

ful model in CF. Given an incomplete rating matrix $Y \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$ with each entry $Y_{ij} = y_{ij}$, $\forall\,(i,j) \in \mathcal{D}$, the model assumes that the rating matrix Y is low-rank and solves the following optimization problem:

$$\hat{\Theta} = \underset{\Theta\in\mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}}{\operatorname{argmin}} \frac{1}{|\mathcal{D}|} \sum_{(i,j)\in\mathcal{D}} \ell(y_{ij}, \Theta_{ij}) + \lambda\|\Theta\|_*, \quad (12)$$

where $\hat{\Theta} \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$ is a predicted low-rank rating matrix, $\ell(\cdot,\cdot)$ is a loss function such as square loss, and λ is the regularization parameter.

We apply Corollary 3 to Eq.(12) to obtain sample-based explanations. We represent each training pair (i,j) by a matrix $X \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{V}|}$ which contains only one nonzero entry $X_{ij} = 1$, so that $\langle X, \Theta\rangle_F = \Theta_{ij}$. The resulting theorem is as below:

Corollary 4. *(high-dim representers for matrix factorization) Let $\hat{\Theta}$ be the minimizer of Eqn.(12) with rank$(\hat{\Theta}) = k$. Let $U\Sigma V^\top = \hat{\Theta}$ be its full SVD decomposition. For any test sample (i', j') with $1 \le i' \le |\mathcal{U}|$ and $1 \le j' \le |\mathcal{I}|$, its prediction can be decomposed according to training samples:*

$$\hat{\Theta}_{i'j'} = \sum_{i:(i,j')\in\mathcal{D}} -\frac{1}{\lambda|\mathcal{D}|}\ell'(y_{ij}, \hat{\Theta}_{ij})\langle\sqrt{\Sigma}U_i, \sqrt{\Sigma}U_{i'}\rangle \quad (13)$$

$$= \sum_{j:(i',j)\in\mathcal{D}} -\frac{1}{\lambda|\mathcal{D}|}\ell'(y_{ij}, \hat{\Theta}_{ij})\langle\sqrt{\Sigma}V_j, \sqrt{\Sigma}V_{j'}\rangle, \quad (14)$$

where $\sqrt{\Sigma} = diag[\sqrt{\Sigma_{11}}, \cdots, \sqrt{\Sigma_{kk}}]$, $U_i \in \mathbb{R}^{k\times 1}$ and $V_j \in \mathbb{R}^{k\times 1}$ denote i^{th} and j^{th} row of U and V respectively.

Corollary 4 shows that the predicted score between user i' and item j', $\hat{\Theta}_{i'j'}$, can be represented as the sum of attributions to each observed interaction $(i,j) \in \mathcal{D}$. Specifically, Eqn.(13) decomposes predictions according to other users interacted with the same item j', while Eqn.(14) decomposes predictions according to other items interacted with the same user i', They are referred to as *user-based attributions* and *item-based attributions*, respectively. Also, we can observe that a test sample (i', j') is only relevant to training samples with the same user i' or the same item j'. Combining the two attributions, we define the importance score of each training data to a test sample (i', j') as follows:

Definition 1. *(high-dim representers for CF) The importance of a training point $(i,j) \in \mathcal{D}$ to a test sample (i', j'), $\mathbf{I}((i,j),(i',j'))$, is given by*

$$\begin{cases} -\frac{1}{\lambda|\mathcal{D}|}\ell'(y_{ij}, \langle\tilde{U}_i, \tilde{V}_j\rangle)\ \langle\tilde{U}_i, \tilde{U}_{i'}\rangle & \text{if } j = j'. \\ -\frac{1}{\lambda|\mathcal{D}|}\ell'(y_{ij}, \langle\tilde{U}_i, \tilde{V}_j\rangle)\ \langle\tilde{V}_j, \tilde{V}_{j'}\rangle & \text{if } i = i'. \\ 0 & \text{otherwise.} \end{cases} \quad (15)$$

where $\tilde{U} = U\sqrt{\Sigma}$ and $\tilde{V} = V\sqrt{\Sigma}$ are normalized embedding matrices for user and item respectively.

Note that we replace $\hat{\Theta}_{ij}$ with $\langle\tilde{U}_i, \tilde{V}_j\rangle$ as they are equivalent. If a training sample (i,j) has a large importance score, three conditions must be satisfied: (1) It has the same user or item as the test sample. (2) $|\ell'(y_{ij}, \langle\tilde{U}_i, \tilde{V}_j\rangle)|$ must be large. When the loss function $\ell(\cdot,\cdot)$ is strongly convex, it implies that the training sample incurs a large loss. (3) Their normalized user (or item) embeddings are close.

5.2. General Matrix-factorization-based Models

Instead of using the nuclear norm, many matrix factorization methods directly reparameterizing the rating matrix Θ with the product of two low-rank matrices U and V (Koren et al., 2009; Mao et al., 2021), corresponding to user and item embeddings. They then directly solve the following optimization problem:

$$\hat{U}, \hat{V} = \underset{U\in\mathbb{R}^{|\mathcal{U}|\times k}, V\in\mathbb{R}^{|\mathcal{I}|\times k}}{\operatorname{argmin}} \sum_{(i,j)\in\mathcal{D}} \ell(y_{ij}, \langle U_i, V_j\rangle), \quad (16)$$

where the loss function $\ell(\cdot,\cdot)$ is point-wise, and training data $\mathcal{D}$ may include negative samples for implicit CF. Popular choices include binary cross-entropy (BCE) (He et al., 2017), mean square error (MSE) (Fang et al., 2020), and triplet loss (Dahiya et al., 2021).

Theorem 3 does not apply to this formulation since it does not have nuclear norm regularization. While it is possible to replace $UV^\top$ with Θ and retrain the model with nuclear norm regularization, the retrained model may behave differently compared to the given model. However, the formulation does enforce hard low-rank constraints on the rating matrix through reparameterization. Therefore, to conduct sample-based attribution, we assume Eqn.(16) is implicitly regularized and use Definition 1 to obtain the high-dimensional representer. For this formulation, we drop the constant term, $1/\lambda|\mathcal{D}|$, since λ is unavailable and does not affect relative importance among training samples. The process of computing the high-dimensional representer for CF and its time complexity analysis are provided in Section F in the supplementary material.

5.3. Two-tower models

Two-tower networks are widely used in deep recommender systems (Hidasi et al., 2015; Covington et al., 2016; Mao et al., 2021; Li et al., 2019). They encode user information and item information with two separate neural networks, which are called towers. The user tower maps each user (e.g., user history, features, and id) to a k-dimensional user embedding, while the item tower maps each item (e.g., product description and id) to the same embedding space. The prediction score is then calculated by the inner product of the user and item embeddings. Formally, let the two separate towers be f_{θ_1} and g_{θ_2}. The training objective function

can be written as:

$$\hat{\theta}_1, \hat{\theta}_2 = \underset{\theta_1,\theta_2}{\operatorname{argmin}} \sum_{(i,j)\in\mathcal{D}} \ell(y_{ij}, \langle f_{\theta_1}(u_i), g_{\theta_2}(v_j)\rangle), \quad (17)$$

where u_i and v_j denote features of user i and item j. Again, we focus on models trained with point-wise loss functions.

To explain two-tower models, we consider the final interaction layers as a bilinear matrix factorization model and the remaining layers as fixed feature encoders. Then we apply the same explanation technique as MF models to explain them. Specifically, we concatenate embeddings of all users and items to form a user matrix and an item matrix, i.e.

$$\hat{U} = [f_{\hat{\theta}_1}(u_1); \cdots; f_{\hat{\theta}_1}(u_{|\mathcal{U}|})] \in \mathbb{R}^{|\mathcal{U}|\times k}$$
$$\text{and } \hat{V} = [g_{\hat{\theta}_2}(v_1); \cdots; g_{\hat{\theta}_2}(v_{|\mathcal{I}|})] \in \mathbb{R}^{|\mathcal{I}|\times k}. \quad (18)$$

Then we use Definition 1 to obtain its sample-based explanations.

6. Experimental Results

We perform experiments on multiple datasets to validate that the proposed method is a preferable choice compared with other sample-based explanation methods such as ℓ_2 representer point selection and influence function, under the high dimensional setting. Moreover, we showcase the utility of the high-dimensional representer in understanding predictions of recommender systems. We also provide another use case for improving negative sampling strategies for collaborative filtering in Appendix C and additional comparisons with other approaches in Appendix G.

6.1. Evaluation Metrics

For quantitative evaluations, we use *case deletion diagnostics* (Yeh et al., 2022; Han et al., 2020; Cook & Weisberg, 1982) as our primary evaluation metric. This metric measures the difference in models' prediction score at a particular test sample z' after removing (a group of) influential training samples and retraining whole models. This metric helps validate the efficacy of sample-based explanation methods and provides a quantitative measurement.

We denote two metrics as $\text{DEL}_+(z', k, \mathbf{I})$ and $\text{DEL}_-(z', k, \mathbf{I})$ separately. These two metrics measure *the difference between models' prediction scores when we remove top-k positive (negative) impact samples given by method* $\mathbf{I}$ *and the prediction scores of the original models.* We expect DEL_+ to be negative and DEL_- to be positive since models' prediction scores should decrease (increase) when we remove positive (negative) impact samples.

To evaluate deletion metric at different k, we follow Yeh et al. (2022) and report area under the curve (AUC):

$$\text{AUC-DEL}_+ = \sum_{i=1}^{m} \frac{\text{DEL}_+(z', k_i, \mathbf{I})}{m}, \text{AUC-DEL}_- = \sum_{i=1}^{m} \frac{\text{DEL}_-(z', k_i, \mathbf{I})}{m},$$

where $k_1 < k_2 < \cdots < k_m$ is a predefined sequence of k.

6.2. Quantitative Evaluation on ℓ_1-regularized Models

In this section, we evaluate the effectiveness of the high-dimensional representer in explaining ℓ_1-regularized logistic regression.

6.2.1. Experimental Settings

Datasets and models being explained: We use the following three datasets on binary classification. **(1) 20 newsgroups**[1]**:** This dataset contains roughly $20,000$ newsgroups posts on 20 topics. It contains $19,996$ samples with $1,355,191$ features. We randomly split 10% data for the test set. **(2) Gisette (Guyon et al., 2004):** It is a handwritten digit recognition problem, which contains highly confusible digits '4' and '9'. It contains $6,000/1,000$ samples with each containing $5,000$ features for training/testing. **(3) Rcv1 (Lewis et al., 2004):** It is a benchmark dataset on text categorization. It has $20,242/677,399$ samples for training/testing. We use bag-of-words features with dimensions $47,236$. We train logistic regression models with ℓ_1 regularization using LIBLINEAR (Fan et al., 2008) on the three datasets. The accuracy of models on the three datasets is above 97%.

Baselines: We compare the high-dimensional representer with the ℓ_2 representer, the influence function (IF) and random deletions. Given a test sample x', the ℓ_2 representer calculates importance score of a training point (x_i, y_i) to the test sample x' with the following formula:

$$\mathbf{I}_{\ell_2}((x_i, y_i), x') = -\ell'(y_i, \langle x_i, \hat{\theta}\rangle))\langle x_i, x'\rangle.$$

For the influence function, we adopt the formula in Proposition 5.3 of Avella-Medina (2017). Assume only the first $q \le p$ entries of the minimizer $\hat{\theta}$ are nonzero, the influence function, $\mathbf{I}_{IF}((x_i, y_i), x')$, is given by

$$-(\frac{1}{n}\nabla_{\theta_{1:q}}\ell(y_i, \langle x_i, \hat{\theta}\rangle) + \lambda \text{sign}(\hat{\theta})_{1:q})^\top H^{-1}_{\hat{\theta}_{1:q}} x'_{1:q},$$

where $H_{\hat{\theta}_{1:q}} = \sum_{i=1}^{n} \nabla^2_{\theta_{1:q}}\ell(y_i, \langle x_i, \hat{\theta}\rangle) \in \mathbb{R}^{q\times q}$. The calculation of the influence function can be simply viewed as first projecting features x, x', and the model parameter $\hat{\theta}$ to nonzero entries of $\hat{\theta}$ and then computing the influence function normally. Notice that the naive implementation takes $O(nq^3 + np)$ time complexity to compute inverse hessian matrix, while the high-dim and ℓ_2 representers only take

[1] http://qwone.com/ jason/20Newsgroups/

$O(np)$ to compute importance scores of all training samples to a test prediction.

To compute AUC-DEL scores, we set $k_i = 0.01iN$ for $1 \leq i \leq 5$. We remove 1% to 5% of positive (negative) impact training samples and report the averaged prediction difference after removing these samples. Each metric is reported over 40 trials with each trial containing 40 test samples.

6.2.2. Results

The results of the four methods are presented in Table 1. We also report the averaged runtime of computing the importance of one test prediction to all training data on a single CPU. The results show that the high-dimensional representer outperforms the other three methods and is over 25x faster than the influence function. Also, the ℓ_2 representer is slightly faster than the high-dimensional representer since inner product is fast when the training data is sparse, and the high-dimensional representer requires one extra step to project vectors to low-dimensional model subspace.

Datasets	20 newsgroups	Gisette	Rcv1
AUC-DEL$_+$			
High-dim Rep.	$\mathbf{-3.733} \pm 0.093$	$\mathbf{-1.000} \pm 0.081$	$\mathbf{-3.208} \pm 0.060$
ℓ_2 Rep.	-2.472 ± 0.067	-0.577 ± 0.073	-2.780 ± 0.057
IF	-2.583 ± 0.043	-0.531 ± 0.011	-2.652 ± 0.040
Random	0.006 ± 0.014	0.010 ± 0.022	0.009 ± 0.005
AUC-DEL$_-$			
High-dim Rep.	$\mathbf{7.478} \pm 0.194$	$\mathbf{3.116} \pm 0.110$	$\mathbf{3.170} \pm 0.077$
ℓ_2 Rep.	5.214 ± 0.143	2.118 ± 0.093	2.726 ± 0.067
IF	4.894 ± 0.086	0.523 ± 0.013	3.065 ± 0.082
Random	0.003 ± 0.014	0.007 ± 0.024	0.007 ± 0.005
Runtime (ms)			
High-dim Rep.	61.35 ± 0.59	$\mathbf{87.34} \pm 0.71$	10.61 ± 0.13
ℓ_2 Rep.	$\mathbf{59.47} \pm 0.58$	130.16 ± 0.34	$\mathbf{6.14} \pm 0.22$
IF	2678.38 ± 3.19	3628.70 ± 2.007	263.90 ± 1.01

Table 1. Case deletion diagnostics for removing positive (negative) impact training samples on various datasets and models and run time comparison. 95% confidence interval of averaged deletion diagnostics on $40 \times 40 = 1,600$ samples is reported. Averaged runtimes over 100 samples are also reported. Smaller (larger) AUC-DEL$_+$ (AUC-DEL$_-$) is better.

6.3. Quantitative Evaluation on Collarborative Filtering

In this section, we evaluate the effectiveness of the high-dimensional representer on explaining CF models in recommender systems.

6.3.1. Experimental Settings

Datasets: (1) Movielens-1M (Harper & Konstan, 2015): It contains about 1M ratings (1-5) from 6,040 users on 3,706 movies. **(2) Amazon review (2018) (Ni et al., 2019):** This dataset contains reviews and ratings (1-5) of products on Amazon. Since the whole dataset is too large, we use data in the video games category, which contains 284,867 ratings from 15,517 users to 37,077 items. We follow the preprocessing procedure in Cheng et al. (2019). We filter out users and items with less than 10 interactions. For every user, we randomly held out two items' ratings to construct the validation and test sets. Also, we normalize all ratings to $[-1, 1]$.

Models being explained: We test the high-dimensional representer on three different models: (1) Matrix factorization with nuclear norm regularization (MF w. nuclear norm) as in Eqn.(7). We do not run this model on Amazon review dataset because the rating matrix is too large. (2) Matrix Factorization (MF) as in Eqn.(16). (3) YoutubeNet (Covington et al., 2016), which uses a deep neural network to encode user features and is one of the representative deep two-tower models.

All models are trained with squared loss. We use soft-impute (Mazumder et al., 2010) algorithm to train the Model (1). The models (2) and (3) are optimized by stochastic gradient descent. Hyper-parameters and model structures are detailed in Appendix E.

Baselines: We compare the high-dimensional representer with the following three baselines: (1) Fast influence analysis (FIA): since the influence function is not scalable to the size of common recommender system benchmarks, Cheng et al. (2019) propose FIA as an approximation of the influence function for MF-based models. (2) Random deletion, which we randomly delete training samples with the same user or item as the given test sample.

Notice that the FIA are not applicable to the MF with nuclear norm model since it is only applicable to MF models in Eqn.(16). Also, the ℓ_2 representer is not applicable to models with two separate encoders since it cannot be treated as a linear mapping. We leave the comparison to TracIn (Pruthi et al., 2020), which is only applicable to models trained with SGD-based optimizers, to the supplementary material.

6.3.2. Setup

We combine user-based and item-based explanations and sort them according to their importance scores. For MovieLens-1M, we drop $k = 10, 20, 30, 40, 50$ samples. For Amazon reviews, we drop $k = 3, 6, 9, 12, 15$ samples. Each metric is averaged over 40 trials with each trial having 40 test samples.

6.3.3. Results

Table 2 summarises the results of different methods. First, we observe that randomly removing samples have roughly no/negative effects on models' predictions for MovieLens-1M/Amazon reviews, and all other methods outperform the

Datasets	Models	Metrics	Methods		
			High-dim Rep.	FIA	Random
MovieLens-1M	MF w. nuclear norm	AUC-DEL$_+$	$\mathbf{-0.225} \pm 0.006$	-	-0.002 ± 0.002
		AUC-DEL$_-$	$\mathbf{0.160} \pm 0.004$	-	-0.002 ± 0.002
	MF	AUC-DEL$_+$	$\mathbf{-0.196} \pm 0.006$	-0.101 ± 0.004	-0.002 ± 0.002
		AUC-DEL$_-$	$\mathbf{0.169} \pm 0.004$	0.072 ± 0.004	-0.001 ± 0.002
	Youtube-Net	AUC-DEL$_+$	$\mathbf{-0.227} \pm 0.008$	-0.096 ± 0.006	-0.001 ± 0.004
		AUC-DEL$_-$	$\mathbf{0.214} \pm 0.007$	0.113 ± 0.007	0.006 ± 0.004
Amazon reviews 2018 (video games)	MF	AUC-DEL$_+$	$\mathbf{-0.184} \pm 0.012$	-0.123 ± 0.011	-0.070 ± 0.011
		AUC-DEL$_-$	$\mathbf{0.080} \pm 0.012$	-0.009 ± 0.012	-0.077 ± 0.011
	Youtube-Net	AUC-DEL$_+$	$\mathbf{-0.234} \pm 0.014$	-0.056 ± 0.013	-0.032 ± 0.011
		AUC-DEL$_-$	$\mathbf{0.294} \pm 0.011$	0.069 ± 0.013	-0.032 ± 0.011

Table 2. Case deletion diagnostics for removing positive (negative) impact training samples on various datasets and models. 95% confidence interval of averaged deletion diagnostics on $40 \times 40 = 1,600$ samples is reported. Smaller (larger) AUC-DEL$_+$ (AUC-DEL$_-$) is better.

random deletiton baseline. Second, the high-dimensional representer outperforms FIA and random deletion in all settings, indicating that the high-dimensional representer is able to estimate the importance of each training sample more accurately.

6.4. Explaining Recommender Systems' Predictions

Movies	User's rating	Movie genre	Importance
Men in Black	3	Action,Sci-Fi, Comedy,Adventure	-4.55
Diabolique	2	Drama/Thriller	-4.03
Independence Day (ID4)	5	Action,Sci-Fi, War	3.52
Star Trek IV: The Voyage Home	5	Action,Sci-Fi, Adventure	3.12
Star Trek V: The Final Frontier	2	Action,Sci-Fi, Adventure	-2.86
Star Trek: First Contact	5	Action,Sci-Fi, Adventure	2.59

Table 3. An example of item-based explanations. In the example, a MF model predicts one user's rating for the movie "Star Trek VI: The Undiscovered Country" to be 3.89. The genres of the movie are action, sci-fi, and adventure.

In this section, we show that the high-dimensional representer generates explanations based on users' historically interacted products for collaborative filtering models.

Table 3 shows an example of an explanation for movie recommendations. We use an MF model trained with square loss to predict users' ratings from 1 to 5 on Movielens-100k, a smaller version of Movielens-1M. We first choose a user with 87 historical ratings and predicts their rating on "Star Trek VI: The Undiscovered Country", calculate similarity scores with the high-dimensional representer on the user's past ratings, and then sort the items according the absolute importance scores. The explanation can be interpreted as "the MF model predicts your rating on *Star Trek VI: The Undiscovered Country* to be 3.89 mostly because of your ratings on the following six movies."

The explanation consists of movies with similar genres and prequels of "Star Trek VI". We see that the model learns the relations of movies from the explanation since movie names and genres are not provided during training. Also, the user's past ratings of 2 or 3 negatively impact the prediction, and ratings of 5 have positive influence. It is reasonable since the user's preference for similar movies would impact the model's predicted ratings. Notice that the high-dimensional representer can also be used to provide user-based explanations in terms of the influence of other users' ratings on the same movie. We do not show these explanations here since user information is lacking in most publicly available datasets. More examples can be found in Appendix D.

7. Conclusion

In this paper, we present *high-dimensional representers* to explain predictions of high-dimensional models in terms of contributions from each of the training samples. We investigate its consequences for canonical instances of sparse models, as well as low-rank models, together with a case study on collaborative filtering, which we consider low-rank matrix-factorization-based models as well as their deep neural variants. In future work, it would be of interest to derive corollaries of our general result for additional instances of high-dimensional models such as group-structured models, as well as additional applications such compressed sensing and sparse Gaussian graphical model estimation.

References

Avella-Medina, M. Influence functions for penalized m-estimators. *Bernoulli*, 23(4B):3178–3196, 2017.

Bae, J., Ng, N., Lo, A., Ghassemi, M., and Grosse, R. If influence functions are the answer, then what is the question? *arXiv preprint arXiv:2209.05364*, 2022.

Bohn, B., Rieger, C., and Griebel, M. A representer theorem for deep kernel learning. *The Journal of Machine Learning Research*, 20(1):2302–2333, 2019.

Brophy, J., Hammoudeh, Z., and Lowd, D. Adapting and evaluating influence-estimation methods for gradient-boosted decision trees. *arXiv preprint arXiv:2205.00359*, 2022.

Candes, E. and Recht, B. Exact matrix completion via convex optimization. *Communications of the ACM*, 55 (6):111–119, 2012.

Candes, E. J. and Recht, B. Exact low-rank matrix completion via convex optimization. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, pp. 806–812. IEEE, 2008.

Candès, E. J. and Tao, T. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–2080, 2010.

Chen, Y., Li, B., Yu, H., Wu, P., and Miao, C. Hydra: Hypergradient data relevance analysis for interpreting deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 7081–7089, 2021.

Cheng, W., Shen, Y., Huang, L., and Zhu, Y. Incorporating interpretability into latent factor models via fast influence analysis. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 885–893, 2019.

Cook, R. D. and Weisberg, S. Characterizations of an empirical influence function for detecting influential cases in regression. *Technometrics*, 22(4):495–508, 1980.

Cook, R. D. and Weisberg, S. *Residuals and influence in regression*. New York: Chapman and Hall, 1982.

Covington, P., Adams, J., and Sargin, E. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pp. 191–198, 2016.

Dahiya, K., Agarwal, A., Saini, D., Gururaj, K., Jiao, J., Singh, A., Agarwal, S., Kar, P., and Varma, M. Siamesexml: Siamese networks meet extreme classifiers with 100m labels. In *International Conference on Machine Learning*, pp. 2330–2340. PMLR, 2021.

Ding, J., Quan, Y., Yao, Q., Li, Y., and Jin, D. Simplify and robustify negative sampling for implicit collaborative filtering. *Advances in Neural Information Processing Systems*, 33:1094–1105, 2020.

Donoho, D. L. Compressed sensing. *IEEE Transactions on information theory*, 52(4):1289–1306, 2006.

Eldar, Y. C. and Kutyniok, G. *Compressed sensing: theory and applications*. Cambridge university press, 2012.

Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. Liblinear: A library for large linear classification. *the Journal of machine Learning research*, 9: 1871–1874, 2008.

Fang, M., Gong, N. Z., and Liu, J. Influence function based data poisoning attacks to top-n recommender systems. In *Proceedings of The Web Conference 2020*, pp. 3019–3025, 2020.

Feldman, V. and Zhang, C. What neural networks memorize and why: Discovering the long tail via influence estimation. *Advances in Neural Information Processing Systems*, 33:2881–2891, 2020.

Friedman, J., Hastie, T., and Tibshirani, R. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

Ghorbani, A. and Zou, J. Data shapley: Equitable valuation of data for machine learning. In *International Conference on Machine Learning*, pp. 2242–2251. PMLR, 2019.

Guo, H., Rajani, N. F., Hase, P., Bansal, M., and Xiong, C. Fastif: Scalable influence functions for efficient model interpretation and debugging. *arXiv preprint arXiv:2012.15781*, 2020.

Guyon, I., Gunn, S., Ben-Hur, A., and Dror, G. Result analysis of the nips 2003 feature selection challenge. *Advances in neural information processing systems*, 17, 2004.

Halko, N., Martinsson, P.-G., and Tropp, J. A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, 2011.

Hammoudeh, Z. and Lowd, D. Identifying a training-set attack's target using renormalized influence estimation. *arXiv preprint arXiv:2201.10055*, 2022.

Han, X., Wallace, B. C., and Tsvetkov, Y. Explaining black box predictions and unveiling data artifacts through influence functions. *arXiv preprint arXiv:2005.06676*, 2020.

Harper, F. M. and Konstan, J. A. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pp. 173–182, 2017.

He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., and Wang, M. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pp. 639–648, 2020.

Hidasi, B., Karatzoglou, A., Baltrunas, L., and Tikk, D. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

Hsieh, C.-J. and Olsen, P. Nuclear norm minimization via active subspace selection. In *International Conference on Machine Learning*, pp. 575–583. PMLR, 2014.

Jaggi, M. and Sulovskỳ, M. A simple algorithm for nuclear norm regularized problems. In *ICML*, 2010.

Jia, R., Dao, D., Wang, B., Hubis, F. A., Gurel, N. M., Li, B., Zhang, C., Spanos, C. J., and Song, D. Efficient task-specific data valuation for nearest neighbor algorithms. *arXiv preprint arXiv:1908.08619*, 2019.

Keerthi, S. S., DeCoste, D., and Joachims, T. A modified finite newton method for fast solution of large scale linear svms. *Journal of Machine Learning Research*, 6(3), 2005.

Khanna, R., Kim, B., Ghosh, J., and Koyejo, S. Interpreting black box predictions using fisher kernels. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 3382–3390. PMLR, 2019.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In *International conference on machine learning*, pp. 1885–1894. PMLR, 2017.

Kong, S., Shen, Y., and Huang, L. Resolving training biases via influence-based data relabeling. In *International Conference on Learning Representations*, 2021.

Koren, Y., Bell, R., and Volinsky, C. Matrix factorization techniques for recommender systems. *Computer*, 42(8): 30–37, 2009.

Kwon, Y. and Zou, J. Beta shapley: a unified and noise-reduced data valuation framework for machine learning. *arXiv preprint arXiv:2110.14049*, 2021.

Lewis, D. D., Yang, Y., Russell-Rose, T., and Li, F. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397, 2004.

Li, C., Liu, Z., Wu, M., Xu, Y., Zhao, H., Huang, P., Kang, G., Chen, Q., Li, W., and Lee, D. L. Multi-interest network with dynamic routing for recommendation at tmall. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 2615–2623, 2019.

Mao, K., Zhu, J., Wang, J., Dai, Q., Dong, Z., Xiao, X., and He, X. Simplex: A simple and strong baseline for collaborative filtering. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pp. 1243–1252, 2021.

Mazumder, R., Hastie, T., and Tibshirani, R. Spectral regularization algorithms for learning large incomplete matrices. *The Journal of Machine Learning Research*, 11: 2287–2322, 2010.

Negahban, S. and Wainwright, M. J. Estimation of (near) low-rank matrices with noise and high-dimensional scaling. *The Annals of Statistics*, 39(2):1069–1097, 2011.

Negahban, S. N., Ravikumar, P., Wainwright, M. J., and Yu, B. A unified framework for high-dimensional analysis of m-estimators with decomposable regularizers. *Statistical science*, 27(4):538–557, 2012.

Ni, J., Li, J., and McAuley, J. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 188–197, 2019.

Pruthi, G., Liu, F., Kale, S., and Sundararajan, M. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 33: 19920–19930, 2020.

Recht, B. A simpler approach to matrix completion. *Journal of Machine Learning Research*, 12(12), 2011.

Schioppa, A., Zablotskaia, P., Vilar, D., and Sokolov, A. Scaling up influence functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8179–8186, 2022.

Schölkopf, B., Herbrich, R., and Smola, A. J. A generalized representer theorem. In *International conference on computational learning theory*, pp. 416–426. Springer, 2001.

Sui, Y., Wu, G., and Sanner, S. Representer point selection via local jacobian expansion for post-hoc classifier explanation of deep neural networks and ensemble models. *Advances in Neural Information Processing Systems*, 34: 23347–23358, 2021.

Thimonier, H., Popineau, F., Rimmel, A., Doan, B.-L., and Daniel, F. Tracinad: Measuring influence for anomaly detection. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6. IEEE, 2022.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Tintarev, N. and Masthoff, J. A survey of explanations in recommender systems. In *2007 IEEE 23rd international conference on data engineering workshop*, pp. 801–810. IEEE, 2007.

Toh, K.-C. and Yun, S. An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems. *Pacific Journal of optimization*, 6(615-640): 15, 2010.

Unser, M. A representer theorem for deep neural networks. *J. Mach. Learn. Res.*, 20(110):1–30, 2019.

Unser, M., Fageot, J., and Gupta, H. Representer theorems for sparsity-promoting l1 regularization. *IEEE Transactions on Information Theory*, 62(9):5167–5180, 2016.

Watson, G. A. Characterization of the subdifferential of some matrix norms. *Linear algebra and its applications*, 170(0):33–45, 1992.

Wojnowicz, M., Cruz, B., Zhao, X., Wallace, B., Wolff, M., Luan, J., and Crable, C. "influence sketching": Finding influential samples in large-scale regressions. In *2016 IEEE International Conference on Big Data (Big Data)*, pp. 3601–3612. IEEE, 2016.

Yang, J., Luo, L., Qian, J., Tai, Y., Zhang, F., and Xu, Y. Nuclear norm based matrix regression with applications to face recognition with occlusion and illumination changes. *IEEE transactions on pattern analysis and machine intelligence*, 39(1):156–171, 2016.

Yeh, C.-K., Kim, J., Yen, I. E.-H., and Ravikumar, P. K. Representer point selection for explaining deep neural networks. *Advances in neural information processing systems*, 31, 2018.

Yeh, C.-K., Taly, A., Sundararajan, M., Liu, F., and Ravikumar, P. First is better than last for training data influence. *arXiv preprint arXiv:2202.11844*, 2022.

Yuan, M. and Lin, Y. Model selection and estimation in the gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.

Zhang, Y., Chen, X., et al. Explainable recommendation: A survey and new perspectives. *Foundations and Trends® in Information Retrieval*, 14(1):1–101, 2020.

A. Overview of the Appendix

The appendix is structured as follows: in Appendix B, we explore the potential social impacts of our work. Additionally, in Appendix C, we utilize high-dimensional representers to enhance negative sampling strategies for collaborative filtering models, which serves as another use case. Furthermore, in Appendix D, we present additional qualitative examples that demonstrate the use of high-dimensional representers in explaining recommender systems. Detailed experimental information regarding the experiments in Section 6 can be found in Appendix E. We also provide the pseudocode and time complexity analysis of the high-dimensinoal representer for collaborative filtering in Appendix F. Moreover, a comparison between the high-dimensional representers for collaborative filtering and TracIn (Pruthi et al., 2020) is presented in Appendix G. Lastly, in Appendix H, we include proofs of our theoretical results.

B. Potential Social Impact of Our Work

One potential social impact is that one may use our approach to change models' predictions via adjusting training samples. This may have positive impacts, such as debugging models or making models fair, and negative impacts, such as attacking existing models or making models more biased and unethical.

C. Another Use Case: Improving Negative Sampling Strategies

In this experiment, we show that high-dimensional representers can be used to improve negative sampling strategies that are widely used to train collaborative filtering models for implicit signals.

Motivation: Implicit CF learns from users' behavior that implicitly affects users' preferences. For example, it may learn from the clicks of users or users' watching history. In this setting, user-item interactions usually contain only positive interactions, and practitioners usually regard all other unobserved interactions as negative samples. However, these unobserved interactions may include false negatives. For instance, users may ignore items not displayed to them, not necessarily because users dislike them. Such false negatives have been demonstrated to be harmful to models (Ding et al., 2020). However, identifying false negatives is challenging since it is impossible to ask users to look over all items and mark their preferences.

Proposed approach: We propose to measure *aggregated importance scores* of negative samples to identify these false negatives. These scores quantitatively measure the extent to which negative pairs contribute to the decrease in prediction scores for observed positive interactions. Larger aggregated importance scores indicate that the negative pair reduces the model's confidence in other known positive interactions, suggesting a higher likelihood of being a false negative.

Let $\mathcal{D} = \mathcal{P} \cup \mathcal{N}$ be the training set comprising positive interactions $\mathcal{P}$ and negative samples $\mathcal{N}$ selected through a negative sampling strategy. The aggregated importance scores are defined as follows:

$$\mathbf{I}_{neg}((i,j)) = \sum_{(i',j') \in \mathcal{P}} \mathbf{I}((i,j),(i',j')), \tag{19}$$

where $\mathbf{I}(\cdot,\cdot)$ is the importance score provided by the high-dimensional representer as in Definition 1. $\mathbf{I}_{neg}((i,j))$ can be interpreted as the sum of importance scores of a negative sample to all positive samples in the training set.

C.1. Experimental Setup

To validate the effectiveness of high-dimensional representers in improving negative sampling strategies, we first train a base model using a normal negative sampling strategy, and then retrain the model after removing identified negative samples. We use the change in the models' performance to measure the performance of the proposed method.

Datasets: We use a binarized MovieLens-100k dataset, which contains $100,000$ ratings (1-5) from 943 users on 1,682 movies. We transform user ratings into binary signals by dropping user ratings less than 4 and treating other interactions as positive samples. In accordance with Toh & Yun (2010); Jaggi & Sulovskỳ (2010), we randomly selected 50% of the ratings for training and the others for the test set.

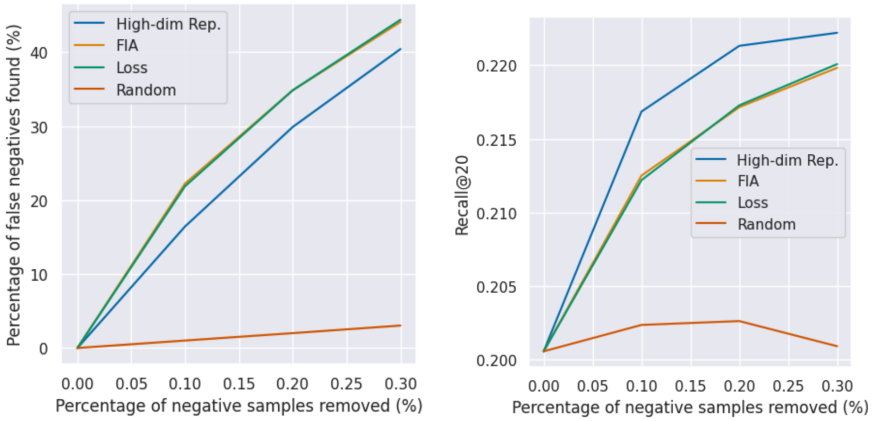
Base models: We first train a matrix factorization model with uniformly selected negative samples. The model is trained with binary cross entropy loss function with the following formulation:

$$\underset{\substack{U \in \mathbb{R}^{|\mathcal{U}| \times k}, \\ V \in \mathbb{R}^{|\mathcal{I}| \times k}}}{\operatorname{argmin}} - \sum_{(i,j) \in \mathcal{P}} \log(\sigma(\langle U_i, V_j \rangle)) - 0.05 \sum_{(i,j) \in \mathcal{N}} \log(1 - \sigma(\langle U_i, V_j \rangle)),$$

where $\sigma(\cdot)$ denotes a sigmoid function, and $\mathcal{N} = \mathcal{D} \backslash \mathcal{P}$ contains all unknown user-item interactions. We multiply loss functions with negative samples with 0.05 since it improves the models' performance. After calculating aggregated importance scores of all negative samples, we remove the top $p\%$ samples with the least scores from $\mathcal{N}$ and train a new MF model with the same objective.

Evaluation metrics: In order to assess the effectiveness of the proposed methods, we utilize the following two evaluation metrics:

1. Number of false negatives identified: Given the impracticality of labeling all user-item interactions, we consider only the positive interactions in the test set as potential false negatives. This metric evaluates the number of false negatives correctly identified by each method.



(a) (b)

Figure 1. (a) Averaged percentage of false negative samples found. (b) Averaged models' performance improvement after removing top negative samples.

2. Performance improvement of the base model after retraining: We measure the change in performance of the base model after removing the top $p\%$ of negative samples identified by each method. The models' performance is evaluated using the recall@20 metric on the test set (He et al., 2020).

These evaluation metrics enable us to assess the ability of the proposed methods to accurately identify false negatives and quantify the improvement achieved in the performance of the base model through retraining.

Baselines: We compare the high-dimensional representer with (1) fast influence analysis, (2) loss functions, and (3) random selections. For FIA, we use importance scores provided by FIA to compute aggregated importance scores in Eqn.(19). For loss functions, we filter out top $p\%$ negative samples with highest loss. For random selection, we randomly remove $p\%$ of negative samples from $\mathcal{N}$.

C.2. Experimental Results

The results of our experiments are presented in Figure 1(a) and Figure 1(b). We observe that the high-dimensional representer, loss functions, and FIA outperform random selection on both evaluation metrics. Notably, while the high-dimensional representer identifies slightly fewer false negatives compared to the loss functions and FIA, it identifies more influential false negatives that contribute the most to performance improvement. These findings indicate that the performance of implicit collaborative filtering can be enhanced by removing harmful samples. As a potential future direction, it would be interesting to explore the integration of the high-dimensional representer into negative sampling procedures.

D. More Qualitative Examples

Table 4 and 5 show qualitative examples of explaining recommender systems' predictions with high-dimensional representers. We use a matrix factorization model trained with square loss to predict users' ratings from 1 to 5 on Movielens-100k datasets.

Movies	User's rating	Movie genre	Importance
Henry V	1	drama, war	-13.57
Cop Land	1	crime, drama, mystery	-11.14
Soul Food	5	drama	-8.67
Independence Day (ID4)	2	action, sci-fi, war	-7.05
Things to Do in Denver when You're Dead	5	crime, drama, romance	6.86
Star Trek IV:The Voyage Home	2	action, sci-fi, adventure	-6.66

Table 4. An example of item-based explanations. In the example, a MF model predicts one user's rating for the movie "Star Wars" to be 4.00. The genres of the movie are action, sci-fi, romance, war, and adventure.

Movies	User's rating	Movie genre	Importance
Terminator	1	action, sci-fi, thriller	-13.48
M*A*S*H	1	comedy, war	-12.81
Fantasia	1	animation, children, musical	-11.25
Psycho	2	horror, romance, thriller	-10.33
Batman	1	action, adventure, crime, drama	-9.31
Gone with the wind	2	drama, romance, war	-7.70

Table 5. An example of item-based explanations. In the example, a MF model predicts one user's rating for the movie "Top gun" to be 3.37. The genres of the movie are action and romance.

E. More Experimental Details

In this section, we provide details of our experiment settings.

E.1. ℓ_1 regularized binary classifiers

We detail data preprocessing for the three datasets we use in Section 6.2, and model hyper-parameters. Dataset statistics and hyper-parameters are listed in Table 6.

For the three text classification datasets in Section 6.2, we convert these multiclass datasets into binary datasets. For 20 newsgroup, we follow the preprocessing procedure in Keerthi et al. (2005) to group the 20 topics into two classes. For Gisette, we use the validation set as the testing set since the labels of the testing set are not available. For Rcv1, we treat CCAT and ECAT as the positive class and treat GCAT and MCAT as the negative class. Instances in both positive and negative classes are removed.

Dataset	# of training samples	# of test samples	feature dimension	$n_{\text{train}}\lambda$	Model Accuracy (%)
20 newsgroup	17,997	1999	1,355,191	10	97.50
Gisette	6,000	1,000	5,000	1	97.50
Rcv1	20,242	677,399	47,236	1	97.50

Table 6. Statistics of text classification datasets, model regularization hyper-parameters, and model accuracy.

E.2. Collaborative Filtering

E.2.1. Dataset statistics

Table 7 shows the statistics of datasets we used in Section 6.3, 6.4 and Appendix C.

Dataset	User #	Items #	Interactions #	Density (%)
MovieLens-100k	943	1,682	$55,375^{\dagger}$	3.49
MovieLens-1M	6,040	3,706	1,000,209	4.47
Amazon review (2018) (Video games)	15,517	37,077	284,867	0.05

Table 7. Statistics of different recommender system datasets. †We remove the ratings that are less or equal to 3, so the number of interactions becomes 55,375.

E.2.2. Implementation Details

To evaluate the models' performance, we follow Mazumder et al. (2010) to use mean absolute error (MAE), which measures the absolute distance between normalized true ratings and predicted ratings. Below we detail the model architectures and their performance on different datasets. All models are trained with square error loss functions.

Matrix factorization with nuclear norm: We use Soft-Impute (Mazumder et al., 2010) to train the models. We set max iterations to 20 and embedding dimension to 12 on the MovieLens-1M dataset. It achieves MAE of 0.54.

Matrix factorization: We use SGD optimizer with learning rate 2.0/15.0 with batch size 3000/3000 to train MF model for 10/10 epochs on MovieLens-1M/Amazon reviews 2018. The model achieves MAE of 0.36/0.46 on MovieLens-1M/Amazon reviews 2018.

YoutubeNet: For MovieLens-1M/Amazon reviews 2018, we use Adam optimizer with learning rate 0.001/0.001 with batch size 3000/3000 to train YoutubeNet for 20/10 epochs. We use an embedding of 64/16 trainable parameters to model user and item information. The user feature encoder consists of 4/3 layers of size 64/16 with 0.2/0.2 dropout probabilities. The item feature encoder contains only item embeddings. It achieves MAE of 0.36/0.42.

F. Computation of the High-dimensional Representer for Collaborative Filtering

The whole process of the computation of high-dimensional representers is shown in Algorithm 1. The first step is pre-processing, which computes normalized user and item embedding matrices through computing SVD of $\hat{U}\hat{V}^\top$. Then we calculate $\ell_1$representers with Definition 1. We note that the shape of $\hat{U}\hat{V}^\top$ is $|\mathcal{U}| \times |\mathcal{I}|$ and it is costly to compute SVD of it since we may have millions of users and items in real-world applications. We propose to decompose the computation into computing SVD of $\hat{U}$ and $\hat{V}$ separately and then combining these smaller matrices of size $k \times k$. We show that this decomposition significantly reduces time complexity in the following analysis.

Note that $(U_1U_3)\Sigma_3(V_1V_3)^\top$ in the pre-processing step is a valid SVD of $\hat{U}\hat{V}^\top$ due to the fact that $(U_1U_3)^\top(U_1U_3) = U_3^\top U_1^\top U_1 U_3 = I_k$ and $(V_1V_3)^\top V_1 V_3 = I_k$.

Algorithm 1 Computation of high-dimensional representers for Collaborative Filtering

Input: A trained user embedding matrix $\hat{U} \in \mathbb{R}^{|\mathcal{U}| \times k}$, a trained item embedding matrix $\hat{V} \in \mathbb{R}^{|\mathcal{I}| \times k}$, a loss function $\ell(\cdot, \cdot)$, training dataset $\mathcal{D}$, and a test sample (i', j').
Preprocessing$(\hat{U}, \hat{V})$:
  $U_1, \Sigma_1', V_1^\top \leftarrow SVD(\hat{U})$.
  $U_2, \Sigma_2', V_2^\top \leftarrow SVD(\hat{V}^\top)$.
  $U_3, \Sigma_3, V_3^\top \leftarrow SVD(\Sigma_1 V_1^\top U_2 \Sigma_2')$.
  $\tilde{U} \leftarrow U_1 U_3 \sqrt{\Sigma_3}$, and $\tilde{V} \leftarrow V_1 V_3 \sqrt{\Sigma_3}$.
  Return: $\tilde{U}, \tilde{V}$
Explain$(\tilde{U}, \tilde{V}, \mathcal{D}, \ell, i', j')$:
  Expls $\leftarrow$ []
  # Computation of user-based explanations:
  $L_u \leftarrow \{(i, j') | (i, j') \in \mathcal{D}\}$
  for $(i, j') \in L_u$ **do**
    score $\leftarrow -\ell'(y_{ij'}, \langle \tilde{U}_i, \tilde{V}_{j'} \rangle \langle \tilde{U}_i, \tilde{U}_{i'} \rangle$.
    Append $((i, j'), \text{score})$ to Expls.
  end for
  # Computation of item-based explanations:
  $L_v \leftarrow \{(i', j) | (i', j) \in \mathcal{D}\}$
  for $(i', j) \in L_v$ **do**
    score $\leftarrow -\ell'(y_{i'j}, \langle \tilde{U}_{i'}, \tilde{V}_j \rangle \langle \tilde{V}_j, \tilde{V}_{j'} \rangle$.
    Append $((i', j), \text{score})$ to Expls.
  end for
  Return: Expls

F.1. Analysis of Time Complexity

In algorithm 1, we use a randomized singular value decomposition algorithm (Halko et al., 2011) to decompose matrices. The SVD algorithm takes an input matrix of size $m \times n$ and outputs its decomposition with rank k using $\mathcal{O}(mn \log(k) + (m+n)k^2)$ operations.

At the pre-processing step, we perform this algorithm three times to decompose matrices of size $|\mathcal{U}| \times k$, $|\mathcal{V}| \times k$ and $k \times k$ to rank-k matrices, which takes $O(\max(|\mathcal{U}|, |\mathcal{V}|)k^2)$ time (assume that $|\mathcal{U}|, |\mathcal{V}| >> k$). On the contrary, if we directly perform SVD on $\hat{U}\hat{V}^\top$, the time complexity would be $O(|\mathcal{U}||\mathcal{V}| \log k)$, which is significantly larger than decomposing three smaller matrices. Also, the matrix multiplications on lines 4 and 5 also take $O(\max(|\mathcal{U}|, |\mathcal{V}|)k^2)$ time. Therefore, the overall time complexity of the pre-processing step is $O(\max(|\mathcal{U}|, |\mathcal{V}|)k^2)$.

At the explanation step, let the average number of interactions a user or an item has is n', i.e., the average sizes of L_u and L_v are n'. The average time complexity of explaining a single test sample is $O(n'k)$ to compute inner products between normalized embeddings of size k.

G. Comparison to TracIn (Pruthi et al., 2020)

TracIn traces the loss change of a given test point during training. When the model being explained is trained with stochastic gradient descent, the loss change at each iteration can be attributed to a single training data we used to update the model. However, it is intractable since we do not know the test sample during training. Pruthi et al. (2020) proposes TracInCP as a practical alternative by measuring gradients similarities only on checkpoints $|\mathcal{T}|$ of the model. TracInCP has the following formulation:

$$\mathbf{I}_{\text{TracInCP}}(z_i, z') = \sum_{t \in \mathcal{T}} \eta^{(t)} \nabla_\theta \mathcal{L}(z_i, \theta^{(t)})^\top \nabla_\theta \mathcal{L}(z', \theta^{(t)}), \tag{20}$$

where $\eta^{(t)}, \theta^{(t)}$ are the learning rate and model parameters at iteration t, and $\mathcal{L}$ is a loss function.

To compare the TracIn with the high-dimensional representer, we perform experiments on collaborative filtering since ℓ_1 regularized linear models in Section 6.2 are usually not trained with stochastic gradient descent. The experimental settings are the same as in Section 6.3.

Datasets	Models	Metrics	Methods		
			High-dim Rep.	TracInCP	Random
MovieLens-1M	MF	AUC-DEL$_+$	-0.196 ± 0.006	$\mathbf{-0.217} \pm 0.007$	-0.002 ± 0.002
		AUC-DEL$_-$	$\mathbf{0.169} \pm 0.004$	0.161 ± 0.005	-0.001 ± 0.002
	Youtube-Net	AUC-DEL$_+$	$\mathbf{-0.227} \pm 0.008$	-0.161 ± 0.008	-0.001 ± 0.004
		AUC-DEL$_-$	$\mathbf{0.214} \pm 0.007$	0.165 ± 0.007	0.006 ± 0.004
Amazon reviews 2018 (video games)	MF	AUC-DEL$_+$	-0.184 ± 0.012	$\mathbf{-0.312} \pm 0.013$	-0.070 ± 0.011
		AUC-DEL$_-$	0.080 ± 0.012	$\mathbf{0.158} \pm 0.012$	-0.077 ± 0.011
	Youtube-Net	AUC-DEL$_+$	-0.234 ± 0.014	$\mathbf{-0.245} \pm 0.016$	-0.032 ± 0.011
		AUC-DEL$_-$	$\mathbf{0.294} \pm 0.011$	0.276 ± 0.011	-0.032 ± 0.011

Table 8. Case deletion diagnostics for removing positive (negative) impact training samples on various datasets and models. 95% confidence interval of averaged deletion diagnostics on $40 \times 40 = 1,600$ samples is reported. Smaller (larger) AUC-DEL$_+$ (AUC-DEL$_-$) is better.

Table 8 shows the results. We observe that although in general TracIn is worse than the high-dimensional representer on YoutubeNet, the performance of tracIn is much better than the high-dimensional representers on vanilla MF models. We argue that TracIn for MF has the same formulation as the high-dimensional representers on any checkpoints, so it can be viewed as an ensemble of high-dimensional representers over trajectories.

By applying TracInCP formula (Eqn.(20)) to the MF objective (Eqn.(16)), the TracInCP formulation is as below: given a training point $(i, j) \in \mathcal{D}$, and a test sample (i', j'), the TracInCP importance scores on MF models are

$$\mathbf{I}_{\text{TracInCP}}((i,j),(i',j')) = \begin{cases} -\sum_{t \in \mathcal{T}} \eta^{(t)} \ell'_{\theta^{(t)}}(y_{ij}, \langle \tilde{U}_i, \tilde{V}_j \rangle) \, \langle \tilde{U}_i, \tilde{U}_{i'} \rangle & \text{if } j = j'. \\ -\sum_{t \in \mathcal{T}} \eta^{(t)} \ell'_{\theta^{(t)}}(y_{ij}, \langle \tilde{U}_i, \tilde{V}_j \rangle) \, \langle \tilde{V}_j, \tilde{V}_{j'} \rangle & \text{if } i = i'. \\ 0 & \text{otherwise.} \end{cases} \tag{21}$$

Therefore, it has the same formulation as the high-dimensional representer for CF in Definition 1 except that TracInCP ensembles over the checkpoints on the trajectories. We note that Eqn.(20) uses the same loss function $\mathcal{L}$ for both training and test samples. However, our evaluation criteria is to find positive (or negative) impact samples, which is not necessarily the triaining loss. Therefore, we replace the test loss $\mathcal{L}$ in Eqn.(20) with prediction score $\langle U_{i'}, V_{j'} \rangle$ and then calculate its gradients.

H. Omitted Proofs

H.1. Proof of Theorem 1

Proof. Since $\hat{\theta}$ is the minimizer of Eqn.(1), there exists a subgradient of the regularization $r(\cdot)$, $g = u_\theta + v$ such that

$$0 = \frac{\partial}{\partial\theta}\left(\frac{1}{n}\sum_{i=1}^{n}\ell(y_i, \langle x_i, \hat{\theta}\rangle) + \lambda r(\theta)\right) = \frac{\partial}{\partial\theta}\left(\frac{1}{n}\sum_{i=1}^{n}\ell(y_i, \langle x_i, \hat{\theta}\rangle)\right) + \lambda g.$$

It implies

$$g = -\frac{1}{n\lambda}\sum_{i=1}^{n}\ell'(y_i, \langle x_i, \hat{\theta}\rangle)x_i.$$

After applying inverse transform $(\partial_\theta r)^+$ of the partial differential on both sides, we have

$$\hat{\theta} = (\partial_\theta r)^+(g) = -\frac{1}{n\lambda}\sum_{i=1}^{n}\ell'(y_i, \langle x_i, \hat{\theta}\rangle)\left(\partial_\theta r)^+ x_i\right).$$

By taking inner product with x' on both sides, we have

$$\langle x', \hat{\theta}\rangle = \sum_{i=1}^{n} -\frac{1}{n\lambda}\ell'(y_i, \langle x_i, \hat{\theta}\rangle))\langle(\partial_{\hat{\theta}} r)^{\frac{+}{2}} x_i, (\partial_{\hat{\theta}} r)^{\frac{+}{2}} x'\rangle.$$

□

H.2. Proof of Corollary 2

Proof. Below is another proof without applying Theorem 1:

Since $\hat{\theta}$ is the minimizer of Eqn.(4), there exists a subgradient of the ℓ_1 norm, $v = \partial\|\hat{\theta}\|_1 \in \mathbb{R}^p$ such that

$$0 = \frac{\partial}{\partial\theta}\left(\frac{1}{n}\sum_{i=1}^{n}\ell(y_i, \langle x_i, \hat{\theta}\rangle) + \lambda\|\hat{\theta}\|_1\right) = \frac{\partial}{\partial\theta}\left(\frac{1}{n}\sum_{i=1}^{n}\ell(y_i, \langle x_i, \hat{\theta}\rangle)\right) + \lambda v,$$

where the ith coordinate of v is $v_i = \text{sign}(\theta_i)$ if $\hat{\theta}_i \neq 0$, and $-1 \leq v_i \leq 1$ if $\hat{\theta}_i = 0$ for all $i \in [p]$. The above equation implies:

$$v = -\frac{1}{n\lambda}\sum_{i=1}^{n}\ell'(y_i, \langle x_i, \hat{\theta}\rangle)x_i.$$

Since we have $\hat{\theta} = |\hat{\theta}| \odot v$, by coordinate-wisely multiplying $|\hat{\theta}|$ on both sides, we have

$$\hat{\theta} = |\hat{\theta}| \odot v = -\frac{1}{n\lambda}\sum_{i=1}^{n}\ell'(y_i, \langle x_i, \hat{\theta}\rangle)(|\hat{\theta}| \odot x_i).$$

By taking an inner product with x' on both sides, we have

$$\langle x', \hat{\theta}\rangle = \sum_{i=1}^{n} -\frac{1}{n\lambda}\ell'(y_i, \langle x_i, \hat{\theta}\rangle))\langle\sqrt{|\hat{\theta}|} \odot x_i, \sqrt{|\hat{\theta}|} \odot x'\rangle.$$

□

H.3. Proof of Corollary3

Proof. According to page 40 in Watson (1992), the subgradient of nuclear norm at $\hat{\Theta}$ is

$$\partial\|\hat{\Theta}\|_* = \{UV^\top + W : W \in \mathbb{R}^{d_1 \times d_2}, \|W\|_2 \leq 1, WV = \mathbf{0}, U^\top W = \mathbf{0}\}.$$

Since $\hat{\Theta}$ is the minmizer of Eqn.(7), we can find a subgradient $UV^T + W$ such that the derivative of Eqn.(7) with respect to Θ at $\hat{\Theta}$ is zero:

$$0 = \frac{1}{n}\sum_{i=1}^{n} \ell'(y_i, \langle X_i, \hat{\Theta}\rangle)X_i + \lambda(UV^\top - W)$$
$$\Rightarrow UV^\top = -\frac{1}{n\lambda}\sum_{i=1}^{n} \ell'(y_i, \langle X_i, \hat{\Theta}\rangle)X_i - W$$

Next, by using the fact that $U^\top U = I_k$ is a identity matrix, we have

$$\begin{aligned}\hat{\Theta} &= U\Sigma V^\top = U\Sigma U^\top U V^\top \\ &= U\Sigma U^\top \left(-\frac{1}{n\lambda}\sum_{i=1}^{n} \ell'(y_i, \langle X_i, \hat{\Theta}\rangle)X_i - W\right) \\ &= \sum_{i=1}^{n}\left(-\frac{1}{n\lambda}\ell'(y_i, \langle X_i, \hat{\Theta}\rangle)\right)\left(U\Sigma U^\top X_i\right) \quad \text{(Using the fact that } U^\top W = \mathbf{0}\text{)}\end{aligned}$$

Similarly, we have

$$\hat{\Theta} = \sum_{i=1}^{n} -\frac{1}{n\lambda}\ell'(y_i, \langle X_i, \hat{\Theta}\rangle_F)\left(X_i V^\top \Sigma V\right).$$

By taking inner product w.r.t a test sample X', we have

$$\begin{aligned}\langle X', \hat{\Theta}\rangle_F &= \sum_{i=1}^{n} -\frac{1}{n\lambda}\ell'(y_i, \langle X_i, \hat{\Theta}\rangle))\langle \sqrt{\Sigma}U^\top X_i, \sqrt{\Sigma}U^\top X'\rangle_F \\ &= \sum_{i=1}^{n} -\frac{1}{n\lambda}\ell'(y_i, \langle X_i, \hat{\Theta}\rangle))\langle X_i V\sqrt{\Sigma}, X' V\sqrt{\Sigma}\rangle_F,\end{aligned}$$

using the fact that $\langle A, B\rangle_F = \text{trace}\left(A^\top B\right)$ and the cyclic property of the trace operator. □

H.4. Proof of Corollary 4

Proof. We first show that the optimization problem in Eqn.(12) is a special case of Eqn.(7). Denote the kth data in $\mathcal{D}$ as (i_k, j_k). Let $X_k \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$ a zero matrix except that the (i_k, j_k) coordinate is 1, and $y_k = Y_{i_k, j_k}$. By plugging the X_k, y_k to Eqn.(7) for $1 \le k \le |\mathcal{D}|$, we recover Eqn.(12).

Let $X' \in \mathbb{R}^{|\mathcal{U}|\times|\mathcal{I}|}$ be a zero matrix except that the (i', j') coordinate is 1. By applying Corollary 3, we have

$$\begin{aligned}\langle X', \hat{\Theta}\rangle &= \hat{\Theta}_{i',j'} \\ &= \sum_{k=1}^{|\mathcal{D}|} -\frac{1}{\lambda|\mathcal{D}|}\ell'(y_k, \langle X_k, \hat{\Theta}\rangle))\langle \sqrt{\Sigma}U^\top X_k, \sqrt{\Sigma}U^\top X'\rangle_F \\ &= \sum_{k=1}^{|\mathcal{D}|} -\frac{1}{\lambda|\mathcal{D}|}\ell'(y_k, \hat{\Theta}_{i_k,j_k}))\text{trace}\left(\sqrt{\Sigma}U^\top X' X_k^\top U\sqrt{\Sigma}\right).\end{aligned}$$

Since $X'X_k$ is a zero matrix if $j_k \neq j'$ and is a zero matrix except that the entry (i_k, i') is one if $j_k = j'$, we have

$$\begin{aligned}\hat{\Theta}_{i',j'} &= \sum_{k=1}^{|\mathcal{D}|} -\frac{1}{\lambda|\mathcal{D}|}\ell'(y_k, \hat{\Theta}_{i_k,j_k}))\mathbb{1}(j_k = j')\langle \sqrt{\Sigma}U_{i'}, \sqrt{\Sigma}U_{i_k}\rangle \\ &= \sum_{i:(i,j')\in\mathcal{D}} -\frac{1}{\lambda|\mathcal{D}|}\ell'(y_{ij}, \hat{\Theta}_{ij})\langle \sqrt{\Sigma}U_i, \sqrt{\Sigma}U_{i'}\rangle.\end{aligned}$$

Similarly, by using Eqn.(11), we have

$$\hat{\Theta}_{i',j'} = \sum_{j:(i',j)\in\mathcal{D}} -\frac{1}{\lambda|\mathcal{D}|}\ell'(y_{ij}, \hat{\Theta}_{ij})\langle\sqrt{\Sigma}V_j, \sqrt{\Sigma}V_{j'}\rangle.$$

□