NET Institute*

www.NETinst.org

Working Paper #09-21

September 2009

Manipulation Robustness of Collaborative Filtering Systems

Benjamin Van Roy and Xiang Yan

Stanford University

* The Networks, Electronic Commerce, and Telecommunications ("NET") Institute, http://www.NETinst.org, is a non-profit institution devoted to research on network industries, electronic commerce, telecommunications, the Internet, "virtual networks" comprised of computers that share the same technical standard or operating system, and on network issues in general.

Manipulation Robustness of Collaborative Filtering Systems

Benjamin Van Roy
Stanford University
bvr@stanford.edu

Xiang Yan
Stanford University
xyan@stanford.edu

October 1, 2009

Abstract

A collaborative filtering system recommends to users products that similar users like. Collaborative filtering systems influence purchase decisions, and hence have become targets of manipulation by unscrupulous vendors. We provide theoretical and empirical results demonstrating that while common nearest neighbor algorithms, which are widely used in commercial systems, can be highly susceptible to manipulation, two classes of collaborative filtering algorithms which we refer to as *linear* and *asymptotically linear* are relatively robust. These results provide guidance for the design of future collaborative filtering systems.

1 Introduction

While the expanding universe of products available via Internet commerce provides consumers with valuable options, sifting through the numerous alternatives to identify desirable choices can be challenging. Collaborative filtering (CF) systems aid this process by recommending to users products desired by similar individuals.

At the heart of a CF system is an algorithm that predicts whether a given user will like various products based on his past behavior and that of other users. Nearest neighbor (NN) algorithms, for example, have enjoyed wide use in commercial CF systems, including those of Amazon, Netflix, and Youtube (Bennett, 2006; Linden et al., 2003; Ryan, 2008). A prototypical NN algorithm stores each user's history, which may include, for instance, his product ratings and purchase decisions. To predict whether a particular user will like a particular product, the algorithm identifies a number of other users with similar histories. A prediction is then generated based on how these so-called neighbors have responded to the product. This prediction could be, for example, a weighted average of past ratings supplied by neighbors.

Because purchase decisions are influenced by CF systems, they have become targets of manipulation by unscrupulous vendors. For instance, a vendor can create multiple online identities and use each to rate his own product highly and competitors' products poorly. As an example, Amazon's CF system was manipulated so that users who viewed a spiritual guide written by a well-known Christian evangelist were subsequently recommended a sex manual for gay men (Olsen,

2002). Although this incident may not have been driven by commercial motives, it highlights the vulnerability of CF systems. The research literature offers further empirical evidence that NN algorithms are susceptible to manipulation (Burke et al., 2005; Lam and Riedl, 2004; Mehta and Nejdl, 2008; Mobasher et al., 2005, 2006; O'Mahony et al., 2004; Sandvig et al., 2007; Zhang et al., 2006).

In order to curb manipulation, one might consider authenticating each user by asking for, say, a credit card number to limit the number of fake identities. This may be effective in some situations. However, in Internet services that do not facilitate financial transactions, such as Youtube, requiring authentication would intrude privacy and drive users away. One might also consider using only customer purchase data, when they are available, as a basis for recommendations because they are likely generated by honest users. Recommendation quality may be improved, however, if higher-volume data such as page views are also properly utilized.

In this paper, we seek to understand the extent to which manipulators can hurt the performance of CF systems and how CF algorithms should be designed to abate their influence. We find that, while NN algorithms can be quite sensitive to manipulation, CF algorithms that carry out predictions based on a particular class of probabilistic models are surprisingly robust. For reasons that we will explain in the paper, we will refer to algorithms of this kind as *linear CF algorithms*.

We find that as a user rates an increasing number of products, the average accuracy of predictions made by a linear CF algorithm becomes insensitive to manipulated data. For instance, even if half of all ratings are provided by manipulators who try to promote half of the products, predictions for users with long histories will barely be distorted, on average. To provide some intuition for why our results should hold, we now offer an informal argument. A robust CF algorithm should learn from its mistakes. In particular, differences between its predictions and actual ratings should help improve predictions on future ratings. A linear CF algorithm generates predictions based on a probability distribution that is a convex combination of two distributions: one that it would learn given only data generated by honest users and one that it would learn given only manipulated data. As a user whose ratings we wish to predict provides more ratings, it becomes increasingly clear which of these two distributions better represents his preferences. As a result, the weight placed on manipulated data diminishes and distortion vanishes.

The main theoretical result of this paper formalizes the above argument. In particular, we will define a notion of distortion induced by manipulators and establish an upper bound on distortion, which takes a particularly simple form:

$$\text{distortion} \leq \frac{1}{n} \ln \frac{1}{1-r}.$$

Here r is the fraction of data that is generated by manipulators and n is the number of products that have already been rated by a user whose future ratings we wish to predict. The bound is very general. First, it applies to all linear CF algorithms. Second, it applies to all manipulation

strategies even if manipulators coordinate their actions and produce data with knowledge of all data generated by honest users. The bound demonstrates that as the number of prior ratings n increases, distortion vanishes. It also identifies the number required to limit distortion to a certain level. This offers guidance for the design of a recommendation system: the system may, for example, assess and inform users about the confidence of each recommendation. The system may also require a new user to rate a set number of products before making recommendations to him. To put this in perspective, consider the following numerical example. Suppose a CF system that accepts binary ratings predicts future ratings correctly 80% of the time in the absence of manipulation. If 10% of all ratings are provided by manipulators, according to our bound, the system can maintain a 75% rate of correct predictions by requiring each new user to rate at least 21 products before receiving recommendations.

To broaden the scope of our analysis, we will also study CF algorithms that behave like linear CF algorithms asymptotically as the size of the training set grows. This class of algorithms, which we refer to as *asymptotically linear*, is more flexible in accommodating modeling assumptions that may improve prediction accuracy. We will establish that a relaxed version of our distortion bound for linear CF algorithms applies to asymptotically linear CF algorithms.

We will also show that our distortion bound does not generally hold for NN algorithms. Intuitively, this is because prediction errors do not always improve the selection of neighbors. In particular, as a user provides more ratings, manipulated data that contribute to inaccurate predictions of his future ratings may remain in the set of neighbors while data generated by honest users may be eliminated from it. As a result, distortion of predictions may not decrease. We will later provide an example to illustrate this.

In addition to theoretical results, this paper provides an empirical analysis using a publicly available set of movie ratings generated by users of Netflix's recommendation system. We produce a distorted version of this data set by injecting manipulated ratings generated using a manipulation technique studied in prior literature. We then compare results from application of three CF algorithms: an NN algorithm, a linear CF algorithm called the kernel density estimation algorithm, and an asymptotically linear CF algorithm called the naive Bayes algorithm. Results demonstrate that while performance of the NN algorithm is highly susceptible to manipulation, those of kernel density estimation and naive Bayes algorithms are relatively robust. In particular, the latter two experience distortions lower than the theoretical bound we provide, whereas the distortion for the former exceeds it by far.

One might also wonder whether manipulation robustness of a CF algorithm comes at the expense of its prediction accuracy. As an example, consider an algorithm that fixes predictions for all ratings to be a constant, without regard to the training data. This algorithm is uninfluenced by manipulation but is likely to yield poor predictions, and is therefore not useful. In our experiments, the accuracy demonstrated by the three algorithms all seems reasonable. This suggests that accuracy of a CF algorithm may be achieved alongside robustness.

Our theoretical and empirical results together suggest that commercial recommendation systems using NN algorithms can be made more robust by adopting approaches that we describe. Note that we are not proposing that real-world systems should implement the specific algorithms we present in this paper. Rather, our analysis highlights properties of CF algorithms that lead to robustness and practitioners may benefit from taking these properties into consideration when designing CF systems.

This paper is organized as follows. In the next section, we discuss some related work. In Section 3, we formulate a simplified model that serves as a context for studying alternative CF algorithms. We then establish results concerning the manipulation robustness of NN, linear, and asymptotically linear CF algorithms in Section 4. In Section 5, we present our empirical study. We make some closing remarks in a final section.

2 Related Work

Early research on CF systems focused on their performance in the absence of manipulation (Breese et al., 1998; Drineas et al., 2002; Herlocker et al., 1999; Kleinberg and Sandler, 2008; Motwani and Vassilvitskii, 2007; Moon and Russell, 2008; Sarwar et al., 2001; Schafer et al., 2001). Almost all work on manipulation robustness has been empirical. For example, Burke et al. (2005); Lam and Riedl (2004); Mehta and Nejdl (2008); Mobasher et al. (2005, 2006); O'Mahony et al. (2004); Sandvig et al. (2007); Zhang et al. (2006) present studies on product ratings made publicly available by Internet commerce sites. In each case, manipulated ratings were injected, and CF algorithms were tested on the altered data sets. The results point out that NN algorithms and their variants are susceptible to manipulation. This line of work identifies an effective manipulation scheme, which is to create multiple identities and with each identity, provide positive ratings on products to be promoted while rating other products in a manner indistinguishable from that of honest users. In Mehta and Nejdl (2008); Mobasher et al. (2006); Zhang et al. (2006), algorithms based on probabilistic latent semantic analysis and principal component analysis were tested. It turns out that these algorithms are asymptotically linear under certain assumptions about the data, and indeed, empirical results in these papers suggest that they are relatively robust to manipulation. These prior results support the conclusions of our work.

To the best of our knowledge, the only prior theoretical work on manipulation robustness of CF algorithms is reported in O'Mahony et al. (2004). This work analyzed an NN algorithm that uses the majority rating among a set of neighbors as the prediction of a user's rating in an asymptotic regime of many users, each of whom rates all products. Manipulators rate as honest users would except on one fixed product. A bound is established on the algorithm's prediction error for this product's rating as a function of the percentage of ratings provided by manipulators. In our work, we do not require users to rate all products and do not constrain manipulators to any particular strategies. Further, we study the performance distortion on average, rather than for a single

product. Finally, a primary contribution of our work is in establishing manipulation robustness of linear and asymptotically linear CF algorithms, which turn out to be superior to NN algorithms in this dimension.

Several researchers have proposed alternative approaches to abating the influence of manipulators. In Resnick and Sami (2007), a mechanism is proposed where users accumulate reputations while providing ratings that are later validated by observed product quality, and a user's influence on ratings predictions is limited by his reputation. In this mechanism, a bound is established on the distortion induced by any finite number of manipulators. In Massa and Avesani (2008); O'Donovan and Smyth (2006), researchers propose leveraging trust relationships among users to weight recommendations and fend off manipulation. Mehta (2007); Mobasher et al. (2007); Sandvig et al. (2007); Williams et al. (2007) suggest detecting manipulated ratings based on their patterns and discounting their impact. Our work complements this growing literature. First, additional sources of information can be integrated into the probabilistic framework that we introduce in this paper to further enhance manipulation robustness. Second, the analytical methods that we develop may be useful for studying the benefits of incorporating such information.

Distortion due to manipulation may also be viewed as a loss of utility in a sequential decision problem induced by errors in initial beliefs. Our analysis is based on ideas similar to those that have been used to study the latter topic, which is discussed in Gossner and Tomala (2008).

More broadly speaking, apart from collaborative filtering, there are other ways to aggregate users' response to products in order to provide recommendations. Research has been performed on the manipulation robustness of these systems as well. To get a flavor of this line of work, see Bhattacharjee and Goel (2007); Dellarocas (2006); Friedman et al. (2007); Miller et al. (2005).

3 Model

We now formulate a simplified model that will serve as a context for assessing performance of alternative CF algorithms. We will first define the product ratings that we work with and then introduce measures of distortion induced by manipulators. For the convenience of the reader, we summarize our mathematical notation in tables in Appendix B.

3.1 Ratings Vectors

In our model, a user selects ratings from a set $\mathbf{S}$. To simplify our discussion, we let $\mathbf{S}$ be a finite subset of $[0, 1]$. For example, $\mathbf{S}$ could be $\{0, 1\}$ with 0 representing a negative rating and 1 representing a positive rating. Note that all the results in this paper can be easily generalized to accommodate any finite set $\mathbf{S}$. There are N products, and a user's type is identified by a vector in $\mathbf{S}^N$. Each nth component of this vector reflects how the user would rate the nth product after inspecting it.

The CF system has access to ratings provided by M identities who have rated products in

the past. The data from each mth identity takes the form of a ratings vector $w^m \in S^N$, where $S = \mathbf{S} \cup \{\circ\}$. Here, an element of $\mathbf{S}$ represents a product rating whereas a circle indicates that a product has not been rated. We refer to $W = (w^1, \ldots, w^M) \in S^{N \times M}$ as the *training data*. This data is used by a CF algorithm to predict future ratings.

Consider a user who is distinct from identities that generated the training data and for whom we will generate recommendations. We will refer to such a user as an *active user*. We will think of a CF algorithm as a function that provides a probability mass function (PMF) $p_{n,x,W}$ over $\mathbf{S}$ for each triplet $(n, x, W) \in \{1, \ldots, N\} \times S^N \times S^{N \times M}$. The PMF represents beliefs about how an active user who has so far provided ratings x would rate product n after inspecting it.

Such an algorithm can be used to guide recommendations; for example, the CF system might recommend to the active user the product he is most likely to rate highly among those that he has not already rated.

3.2 Kullback-Leibler Distortion

To study the influence of manipulation, we consider a situation where a fraction r of the identities are created by manipulators, while the remaining fraction $1-r$ correspond to distinct honest users. We denote the honest ratings vectors by $y^1, \ldots, y^{(1-r)M} \in S^N$ and the manipulated ratings vectors by $z^1, \ldots, z^{rM} \in S^N$. Let $Y = (y^1, \ldots, y^{(1-r)M})$ and $Z = (z^1, \ldots, z^{rM})$ so that the training data is $W = (Y, Z)$.

To assess distortion of predictions made by a CF algorithm, we consider the following thought experiment. A hypothetical active user begins with a ratings vector x^0, with each nth component set to $x^0_n = \ \circ$, and inspects products in an order $\nu = (\nu_1, \ldots, \nu_N) \in \sigma_N$, where σ_N denotes the set of permutations of $\{1, \ldots, N\}$. After inspecting product ν_k, the user rates it by sampling from the PMF $p_{\nu_k, x^{k-1}, Y}$. An updated ratings vector x^k is generated by incorporating this new rating in x^{k-1}. This stochastic process reflects how we would think honest users behave based on the CF algorithm and uncorrupted data set Y. We introduce the following measure of distortion, which we refer to as *Kullback-Leibler (KL) distortion*:

$$d_n^{\text{KL}}(p, \nu, Y, Z) = \frac{1}{n} \sum_{k=1}^{n} \mathrm{E}\left[D\left(p_{\nu_k, x^{k-1}, Y} \,\middle\|\, p_{\nu_k, x^{k-1}, (Y,Z)} \right) \right],$$

where D denotes Kullback-Leibler divergence with the natural log. That is, for any two PMFs p and q over support U, $D(p \,\|\, q) = \sum_{u \in U} p(u) \ln\left(p(u)/q(u)\right)$. This measure of the difference between PMFs is commonly used in information theory.

For each k, the PMF $p_{\nu_k, x^{k-1}, Y}$ represents the prediction that would be made in the absence of manipulators, whereas $p_{\nu_k, x^{k-1}, (Y,Z)}$ is what it becomes as a consequence of manipulation. Hence, $D\left(p_{\nu_k, x^{k-1}, Y} \,\middle\|\, p_{\nu_k, x^{k-1}, (Y,Z)} \right)$ measures the extent to which the manipulated data Z influences the prediction. We take the expectation of this quantity, with x^{k-1} distributed as the CF algorithm

would have predicted if the data set were not corrupted by manipulated data. KL distortion $d_n^{\text{KL}}(p, \nu, Y, Z)$ averages these terms over the first n inspected products.

We can view the generation of each x^k in the preceding thought experiment in the context of a Markov chain. In particular, given p and an ordering ν, we define a Markov chain with state space S^N where the initial state is x^0, and for each $k \geq 1$, state x^k evolves according to:

$$x^k_{\nu_k} \sim p_{\nu_k, x^{k-1}, W},$$

$$x^k_j = x^{k-1}_j, \forall\, j \neq \nu_k.$$

We denote by $P(\cdot \mid p, W, \nu)$ the PMF of x^N implied by p, W, and ν, and drop p from the list of conditioning variables when this does not cause confusion. Clearly, p is a conditional PMF with respect to P. That is, for each n and $\mathbf{s} \in \mathbf{S}$,

$$p_{\nu_n, x^{n-1}, W}(\mathbf{x}_{\nu_n} = \mathbf{s}) = P\left(\mathbf{x}_{\nu_n} = \mathbf{s} \;\middle|\; x^{n-1}, W, \nu\right).$$

As such, a CF algorithm p is completely characterized by a corresponding P and we will use P to refer to a CF algorithm from here on. KL distortion can then be re-written as:

$$d_n^{\text{KL}}(P, \nu, Y, Z) = \frac{1}{n} \sum_{k=1}^{n} \mathrm{E}\left[D\left(P\left(\cdot \;\middle|\; x^{k-1}, Y, \nu\right) \middle\| P\left(\cdot \;\middle|\; x^{k-1}, (Y, Z), \nu\right)\right)\right].$$

3.3 Root-Mean-Squared Distortion

Some algorithms such as NN algorithms generate predictions not in the form of PMFs, but as scalars that may be interpreted as the means of PMFs. For these algorithms, it may be more suitable to measure manipulation impact in terms of *root-mean-squared (RMS) distortion*:

$$d_n^{\text{RMS}}(P, \nu, Y, Z) = \sqrt{\frac{1}{n} \sum_{k=1}^{n} \mathrm{E}\left[\left(\hat{x}_{\nu_k, x^{k-1}, Y} - \hat{x}_{\nu_k, x^{k-1}, (Y,Z)}\right)^2\right]},$$

where $\hat{x}_{\nu_k, x^{k-1}, Y}$ and $\hat{x}_{\nu_k, x^{k-1}, (Y,Z)}$ denote the scalar predictions of $\mathbf{x}_{\nu_k}$ by the algorithm based on ratings history x^{k-1} and data sets Y and (Y, Z), respectively. Note that if the algorithm generates PMFs as predictions, $\hat{x}_{\nu_k, x^{k-1}, Y}$ and $\hat{x}_{\nu_k, x^{k-1}, (Y,Z)}$ would be expectations of $\mathbf{x}_{\nu_k}$ taken with respect to $P\left(\cdot \;\middle|\; x^{k-1}, Y, \nu\right)$ and $P\left(\cdot \;\middle|\; x^{k-1}, (Y, Z), \nu\right)$, respectively. The expectation in the definition of RMS distortion is taken with x^{k-1} distributed as the CF algorithm would have predicted based on Y. RMS distortion may offer a more transparent assessment than KL distortion because the former computes how much scalar predictions change in the same unit as the predictions themselves. RMS distortion is bounded by a function of KL distortion:

$$d_n^{\text{RMS}}(P, \nu, Y, Z) \leq \sqrt{\frac{1}{2} d_n^{\text{KL}}(P, \nu, Y, Z)}.$$

This is shown in Proposition 1 in Appendix A.1 by using Pinsker's inequality and an additional lemma.

3.4 Binary Prediction Distortion

To offer an intuitive interpretation for RMS distortion, we consider a setting where users provide binary ratings and the CF system offers binary predictions based on the PMFs that it generates. That is, we set $\mathbf{S} = \{0, 1\}$. Given training data Y, for a user with ratings history x^{k-1}, the system generates a prediction of $\hat{x}_{\nu_k, x^{k-1}, Y} = 1$ for product ν_k if $P\left(\mathbf{x}_{\nu_k} = 1 \,\middle|\, x^{k-1}, Y, \nu\right) \geq 1/2$ and generates a prediction of $\hat{x}_{\nu_k, x^{k-1}, Y} = 0$ otherwise. Similarly, we denote $\hat{x}_{\nu_k, x^{k-1}, (Y,Z)}$ as the binary prediction based on (Y, Z). We define the following *binary prediction distortion*:

$$d_n^{\mathrm{B}}(P, \nu, Y, Z) = \frac{1}{n} \sum_{k=1}^{n} \left(\Pr\left(\mathbf{x}_{\nu_k} = \hat{x}_{\nu_k, x^{k-1}, Y}\right) - \Pr\left(\mathbf{x}_{\nu_k} = \hat{x}_{\nu_k, x^{k-1}, (Y,Z)}\right) \right),$$

where each x_{ν_k} is distributed according to $p_{\nu_k, x^{k-1}, Y}$ and x^{k-1} is distributed as the CF algorithm would have predicted based on Y. This quantity captures the average decrease in the probability of correct predictions, induced by manipulation. It turns out that binary prediction distortion is bounded by RMS distortion:

$$d_n^{\mathrm{B}}(P, \nu, Y, Z) \leq d_n^{\mathrm{RMS}}(P, \nu, Y, Z). \tag{1}$$

Proved in Proposition 2 in Appendix A.1, this result offers an interpretation of RMS distortion as an upper bound on the drop in the probability of correct predictions in a binary setting.

3.5 Discussion

One might wonder how we arrive at our distortion measures. For instance, regarding the motivating throught experiment, one might question why we sample an active user's ratings from the distribution $P\left(\cdot \,|\, Y, \nu\right)$ rather than from a posterior belief obtained by Bayesian update based on a prior belief and observed ratings x^{k-1}. We do so because posteriors are sometimes difficult to represent and Bayesian updates may be computationally prohibitive. The way we sample effectively uses the CF algorithm to encode the prior and approximate its posterior.

One might also wonder why we choose our particular distortion measures over other candidates. For instance, one option is to consider the top n most desirable products based on predictions, and define as distortion some measure of their quality change due to the manipulated samples. One reason why we prefer KL and RMS distortions is that they are convex functions of predictions while this measure is not. As such, this measure is difficult to analyze. Further, as will be discussed in Section 5.6, in a recent competition of CF algorithms, Netflix uses RMS error to assess their prediction accuracies (Netflix Prize, 2006). This suggests that commercial CF algorithms are typically

designed to minimize convex measures of error. Our choice of distortion measures is in line with this approach.

Another option one might consider is to measure the worst distortion over all products. While it would be attractive to keep this measure small for all manipulation schemes, that may not be achievable. Further, a large worst-case distortion does not necessarily imply that a CF algorithm is not useful. Consider, for instance, a case where many manipulators aim to distort ratings of many different products, and the output of the algorithm is such that the rating of one product is significantly distorted while the rest are not. We might still say that the algorithm is reasonably robust whereas a worst-case measure would indicate otherwise. That said, we note that since KL and RMS distortions characterize average distortions, the robustness results of this paper do not provide guarantees on the distortion of individual products' ratings.

4 Collaborative Filtering Algorithms

In this section, we first introduce the notion of *probabilistic CF algorithms*. We then describe two classes of such algorithms, namely linear and asymptotically linear CF algorithms, and analyze their robustness to manipulation. Finally, we discuss nearest neighbor algorithms and their susceptibility to manipulation.

4.1 Probabilistic Collaborative Filtering Algorithms

A probabilistic CF algorithm carries out predictions based on a probabilistic model of how the training data is generated. We will model training data as being generated in the following way. First, a user type $\mathbf{w}^m \in \mathbf{S}^N$ is sampled i.i.d. from some PMF P^*. Then, $w^m \in S^N$ is sampled from a conditional PMF Q^*, conditioned on $\mathbf{w}^m$, which for each n assigns either $w_n^m = \circ$ or $w_n^m = \mathbf{w}_n^m$. Note that this model allows for dependence between the type of a user and the products he chooses to rate. This accommodates, for example, systems in which users tend to inspect and rate only products that they care for.

We will call a CF algorithm P *probabilistic* if for each W, the distribution $P(\cdot \,|\, W, \nu)$ is independent of ν, in which case we drop ν from the list of conditioning variables. This implies that for each W, the algorithm generates a distribution $P(\cdot \,|\, W)$, and for each k, lets

$$p_{\nu_k, x^{k-1}, W} = P\left(\cdot \,\middle|\, x^{k-1}, W\right).$$

4.2 Linear Collaborative Filtering Algorithms

We say that a probabilistic CF algorithm P is *linear* if for any $W_1 \in S^{N \times M_1}$ and $W_2 \in S^{N \times M_2}$,

$$P\left(\cdot \,|\, (W_1, W_2)\right) = \frac{M_1}{M_1 + M_2} P\left(\cdot \,|\, W_1\right) + \frac{M_2}{M_1 + M_2} P\left(\cdot \,|\, W_2\right).$$

This definition states that the PMF $P(\cdot \mid (W_1, W_2))$ that a linear CF algorithm P generates based on training data (W_1, W_2) is a convex combination of two PMFs: namely, the PMF $P(\cdot \mid W_1)$ that it generates based on W_1 and the PMF $P(\cdot \mid W_2)$ that it generates based on W_2.

We now examine the KL distortion that manipulators can induce on a linear CF algorithm. Consider training data $W = (Y, Z)$ consisting of ratings vectors Y from honest users and Z from manipulators, with the latter making up a fraction r of the training data. The following theorem, which is the main theoretical contribution of this paper, establishes a bound on the resulting KL distortion.

Theorem 1. *Fix the number of products N and let P be a linear CF algorithm. Then, for all M, $r \in \{0, 1/M, \ldots, (M-1)/M\}$, $Y \in S^{N \times (1-r)M}$, $Z \in S^{N \times rM}$, $\nu \in \sigma_N$, and $n \in \{1, \ldots, N\}$,*

$$d_n^{\text{KL}}(P, \nu, Y, Z) \leq \frac{1}{n} \ln \frac{1}{1-r}.$$

This result is proved in Appendix A.2.

Note that the bound only depends on the number of active user ratings n and the fraction of data r generated by manipulators. Hence, it represents a worst case bound over all linear CF algorithms P, the number of products N, the quantity M and values (Y, Z) of the training data, and the order ν in which the active user rates products. This means, for example, that it applies even if manipulators coordinate with each other and select ratings with knowledge of the specific CF algorithm P, the honest ratings Y, and the ordering ν. This also makes the bound relevant for realistic models of how a recommendation system might sequence products for a user; for example, each ν_k could be the product that the CF algorithm predicts as being most desirable among remaining ones after the user has inspected products $\nu_1, \ldots, \nu_{k-1}$.

Note that KL distortion vanishes as the number n of products rated by the active user increases. To develop intuition for why this happens, we now offer an informal argument. Observe that $P(\cdot \mid (Y, Z)) = (1-r)P(\cdot \mid Y) + rP(\cdot \mid Z)$. If $P(\cdot \mid Y)$ is identical to $P(\cdot \mid Z)$, then $P(\cdot \mid (Y, Z))$ is equal to $P(\cdot \mid Y)$ and distortion will be zero. Otherwise, if $P(\cdot \mid Y)$ and $P(\cdot \mid Z)$ are different, as an active user inspects and rates products in the manner that we define, his ratings will tend to be distinguished as sampled from $P(\cdot \mid Y)$ rather than from $P(\cdot \mid Z)$. As such, the influence of Z on predictions diminishes as n grows.

The bound depends on r through the term $\ln(1/(1-r))$. This term captures the dependence of KL distortion on the fraction of data produced by manipulators. As one would expect, this term vanishes when r is set to zero.

To offer an additional interpretation of the bound, we refer to the quantity $nd_n^{\text{KL}}(P, \nu, Y, Z)$ as *total distortion*. It is bounded by $\ln(1/(1-r))$, which is close to r if r is small. This implies that the total distortion incurred by manipulated data is roughly equal to its size as a fraction of the entire data set, when this fraction is small.

As a corollary of Theorem 1 and Proposition 1 we have the following bound on RMS distortion.

Corollary 1. *Fix the number of products N and let P be a linear CF algorithm. Then, for all M, $r \in \{0, 1/M, \ldots, (M-1)/M\}$, $Y \in S^{N\times(1-r)M}$, $Z \in S^{N\times rM}$, $\nu \in \sigma_N$, and $n \in \{1, \ldots, N\}$,*

$$d_n^{\mathrm{RMS}}(P, \nu, Y, Z) \leq \sqrt{\frac{1}{2n} \ln \frac{1}{1-r}}.$$

Figure 1 illustrates how this bound depends on r and n. In particular, the four curves from bottom to top correspond to cases where $r = 0.01, 0.05, 0.1, 0.2$, respectively. The bound can offer useful guidance. For example, it ensures that if an active user has rated 22 products and no more than 10% of the training data is manipulated, then the RMS distortion induced by manipulators is less than 0.05. In a setting where users provide binary ratings and the system generates binary predictions, according to our bound on binary prediction distortion in (1) in Section 3.4, the average probability of correct predictions decreases by at most 0.05. Hence, if a binary CF system predicts ratings correctly 80% of the time in the absence of manipulation, it can maintain this probability at 75% in the presence of manipulation if it requires active users to rate 21 products before receiving recommendations.

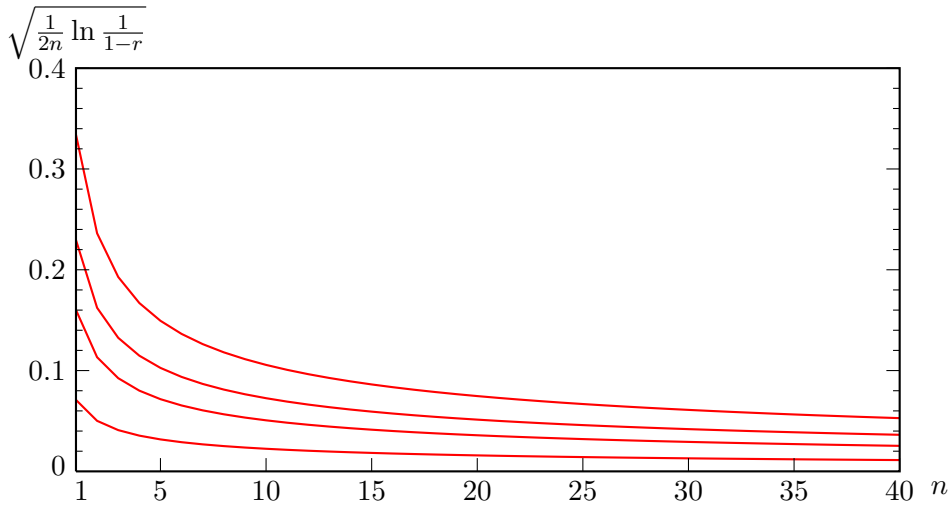


Figure 1: Bound on $d_n^{\mathrm{RMS}}(P, \nu, Y, Z)$ as a function of n. The four curves from bottom to top are for cases where $r = 0.01$, 0.05, 0.1, and 0.2, respectively.

We will introduce examples of linear CF algorithms in Section 5.3.

4.3 Asymptotically Linear Collaborative Filtering Algorithms

We say that a probabilistic CF algorithm P is *asymptotically linear* if for all PMFs of ratings vectors ψ, ϕ over S^N, $r \in [0, 1]$, and $\epsilon > 0$,

$$\lim_{m\to\infty} \Pr\left(D\left(((1-r)P\left(\cdot \mid U_m\right) + rP\left(\cdot \mid V_m\right)) \,\|\, P\left(\cdot \mid (U_m, V_m)\right)\right) \geq \epsilon\right) = 0,$$

where for each m, $U_m = (u^1, \ldots, u^{m-l}) \in S^{N \times (m-l)}$ and $V_m = (v^1, \ldots, v^l) \in S^{N \times l}$, $l \sim \text{Binomial}(m, r)$, and $u^1, \ldots, u^{m-l} \sim \psi$ and $v^1, \ldots, v^l \sim \phi$ are i.i.d. sequences.

To understand the preceding definition, we think of training data (U_m, V_m) for each m as generated in the following way: with probability $1 - r$, a ratings vector is sampled from ψ, which we denote as u^i for an appropriate i, and with probability r, a ratings vector is sampled from ϕ, which we denote as v^j for an appropriate j. U_m consists of all such u^i and V_m consists of all such v^j. As m grows, an asymptotically linear CF algorithm behaves like a linear CF algorithm in that the PMF $P(\cdot \mid (U_m, V_m))$ that it generates based on data (U_m, V_m) converges in probability to a convex combination of two PMFs: namely, the PMF $P(\cdot \mid U_m)$ that it would generate based on U_m and the PMF $P(\cdot \mid V_m)$ that it would generate based on V_m. By an application of the weak law of large numbers, it can be shown that all linear CF algorithms are asymptotically linear.

We can also show that asymptotically linear CF algorithms are asymptotically robust, in a sense to be made precise later. It turns out that this result applies to a broader range of practical algorithms that are asymptotically linear in a more restricted sense, which we now define. Consider a set Ψ of PMFs over S^N. We say that a probabilistic CF algorithm P is *asymptotically linear with respect to* Ψ if for all PMFs ψ, $\phi \in \Psi$, $r \in [0, 1]$, and $\epsilon > 0$,

$$\lim_{m \to \infty} \Pr\left(D\left(((1-r)P(\cdot \mid U_m) + rP(\cdot \mid V_m)) \,\|\, P(\cdot \mid (U_m, V_m))\right) \geq \epsilon\right) = 0,$$

where for each m, $U_m = (u^1, \ldots, u^{m-l}) \in S^{N \times (m-l)}$ and $V_m = (v^1, \ldots, v^l) \in S^{N \times l}$, $l \sim \text{Binomial}(m, r)$, and $u^1, \ldots, u^{m-l} \sim \psi$ and $v^1, \ldots, v^l \sim \phi$ are i.i.d. sequences.

The following theorem and corollary characterize the robustness of asymptotically linear CF algorithms.

Theorem 2. *Fix the number of products N and a set Ψ of PMFs over S^N. Let P be a CF algorithm asymptotically linear with respect to Ψ. Then, for all $\psi, \phi \in \Psi$, $r \in [0, 1)$, $\nu \in \sigma_N$, $n \in \{1, \ldots, N\}$, and $\epsilon > 0$,*

$$\lim_{m \to \infty} \Pr\left(d_n^{\text{KL}}(P, \nu, Y_m, Z_m) \geq \frac{1}{n} \ln \frac{1}{1-r} + \epsilon\right) = 0,$$

where, for each m, $Y_m = (y^1, \ldots, y^{m-l}) \in S^{N \times (m-l)}$, $Z_m = (z^1, \ldots, z^l) \in S^{N \times l}$, $l \sim$ Binomial(m, r), and $y^1, \ldots, y^{m-l} \sim \psi$ and $z^1, \ldots, z^l \sim \phi$ are i.i.d. sequences.

Corollary 2. *Fix the number of products N and a set Ψ of PMFs over S^N. Let P be a CF algorithm asymptotically linear with respect to Ψ. Then, for all $\psi, \phi \in \Psi$, $r \in [0, 1)$, $\nu \in \sigma_N$, $n \in \{1, \ldots, N\}$, and $\epsilon > 0$,*

$$\lim_{m \to \infty} \Pr\left(d_n^{\text{RMS}}(P, \nu, Y_m, Z_m) \geq \sqrt{\frac{1}{2n} \ln \frac{1}{1-r}} + \epsilon\right) = 0,$$

where, for each m, $Y_m = (y^1, \ldots, y^{m-l}) \in S^{N \times (m-l)}$, $Z_m = (z^1, \ldots, z^l) \in S^{N \times l}$, $l \sim$ Binomial(m, r), and $y^1, \ldots, y^{m-l} \sim \psi$ and $z^1, \ldots, z^l \sim \phi$ are i.i.d. sequences.

Theorem 2 is proved in Appendix A.2 and Corollary 2 follows from Theorem 2 and Proposition 1. These results state that for any CF algorithm P asymptotically linear with respect to Ψ and any fixed PMFs ψ, $\phi \in \Psi$, as honest users and manipulators sample more data from them, with high probability, the distortion bounds for linear CF algorithms in Theorem 1 and Corollary 1 will also apply to P and in particular, distortion will vanish as n grows.

The intuition behind these results is similar to that for Theorem 1. In particular, given sufficient data, the learned PMF $P(\cdot \mid (Y_m, Z_m))$ should closely approximate $(1-r)P(\cdot \mid Y_m) + rP(\cdot \mid Z_m)$. If $P(\cdot \mid Y_m)$ and $P(\cdot \mid Z_m)$ are similar, then $P(\cdot \mid (Y_m, Z_m))$ should be close to $P(\cdot \mid Y_m)$ and distortion should be close to zero. On the other hand, if $P(\cdot \mid Y_m)$ and $P(\cdot \mid Z_m)$ are significantly different, as an active user provides more ratings, it will be increasingly clear that they are sampled from $P(\cdot \mid Y_m)$ rather than $P(\cdot \mid Z_m)$, and distortion will diminish.

We now study a class of asymptotically linear CF algorithms, which converge to the true PMF of user types under certain assumptions about the training data. Consider a set Φ of $(\overline{P}, \overline{Q})$ pairs where $\overline{P}$ is a distribution over user types and $\overline{Q}$ is a conditional distribution of ratings vectors conditioned on types. We say that Φ is *identifiable* if each $(\overline{P}, \overline{Q})$ leads to a distinct ratings vector distribution $\overline{PQ}$ over S^N. Given an identifiable set Φ, we say that a probabilistic CF algorithm P is *consistent with respect to* Φ if for all $(\overline{P}, \overline{Q}) \in \Phi$ and $\epsilon > 0$,

$$\lim_{m\to\infty} \Pr\left(D\left(P(\cdot \mid W_m) \,\middle\|\, \overline{P}\right) \geq \epsilon\right) = 0,$$

where for each m, $W_m = (w_1, \ldots, w_m) \in S^{N\times m}$ is generated independently and $w_1, \ldots, w_m \sim \overline{PQ}$ is an i.i.d. sequence. This definition is meant to capture algorithms that converge to type distribution P^* as the training data grows. The following theorem states the setting in which a consistent CF algorithm is asymptotically linear.

Theorem 3. *Any probabilistic CF algorithm consistent with respect to an identifiable and convex set Φ is asymptotically linear with respect to Φ.*

The preceding result, proved in Appendix A.2, together with the definition of consistent algorithms and Theorem 2 imply that if data are sampled i.i.d. from a PMF P^*Q^* induced by (P^*, Q^*) in an identifiable and convex set Φ, then a consistent algorithm with respect to Φ would provide guarantees on both prediction accuracy and robustness to manipulation as training data grows. In practice, even if it is unclear whether the identifiability and convexity conditions hold, as a starting point, one might still apply a consistent CF algorithm, with the hope that it will deliver reasonable accuracy and robustness. In Section 5, we will empirically evaluate a consistent CF algorithm called the naive Bayes algorithm.

4.4 Nearest Neighbor Algorithms

Nearest neighbor algorithms, widely used in commercial CF systems (Bennett, 2006; Linden et al., 2003; Ryan, 2008), generally come in two classes. The first class predicts a user's ratings based on

those provided by similar users, referred to as neighbors. The second class makes predictions on a product based on ratings that the user has provided on similar products, which can also be viewed as neighbors. In this section, we study a simple NN algorithm of the first class and the extent to which its predictions can be distorted by manipulators. We show that the bounds of the previous section do not apply to this NN algorithm, and unlike the case of linear CF algorithms, distortion does not generally diminish as the active user inspects and rates products. Though our analysis focuses on a particular NN algorithm, the resulting insights apply more broadly and in particular, to NN algorithms of the second class as well.

We study the case of binary ratings. NN algorithms identify and weight neighbors using a similarity measure. We will consider a similarity measure that increases by one for each pair of consistent ratings and decreases by one for each pair of inconsistent ratings:

$$s(x,y) = |\{1 \leq n \leq N : x_n = y_n \neq \ \circ\}| - |\{1 \leq n \leq N : \ \circ \neq x_n \neq y_n \neq \ \circ\}|,$$

for any pair of ratings vectors $x, y \in S^N$.

We consider an NN algorithm that predicts the future rating of product n for a user with ratings vector x by carrying out the following steps. First, the algorithm identifies the subset of the training data samples that offer ratings for product n. If this subset is empty, the NN algorithm optimistically predicts a rating of 1. Otherwise, from among these ratings vectors, the ones most similar to x are identified. We denote the resulting set of neighbors, which should be a singleton unless there is a tie, by $\mathcal{N}(n, x, W)$. Finally, an average of their ratings for product n forms the prediction:

$$\hat{x}_{n,x,W} = \frac{\sum_{w \in \mathcal{N}(n,x,W)} w_n}{|\mathcal{N}(n,x,W)|}.$$

Our observations extend to other more complicated similarity metrics and neighbors selection methods. However, we focus on this particular case in order to keep our analysis clean.

We now consider a simple setting that facilitates analysis of RMS distortion in our NN algorithm. We are interested in how RMS distortion changes as the number of ratings n provided by an active user grows. Since n cannot exceed the number of products N, we will define an ensemble of models indexed by N. To facilitate our construction, we will only consider even N.

To keep things simple, we restrict attention to a situation where honest users agree on the ratings of all products. In particular, there is a single user type $\mathbf{x}^{\text{odd}}$ which rates odd-indexed products 1 and even-indexed products 0. The user type PMF P^* assigns all probability to this vector. Each honest ratings vector y^m is generated by sampling a random set of odd numbers between 1 and $N-1$, then for each sample k, replacing components k and $k+1$ of $\mathbf{x}^{\text{odd}}$ with circles. We assume that the honest ratings Y of training data is such that each set of odd numbers between 1 and $N-1$ is sampled exactly once. That is, each element of Y corresponds to an element of the set $\{(1,0),(\circ,\circ)\}^{N/2}$. As such, there are $2^{N/2}$ honest ratings vectors.

Recalling the setting that we use for assessing distortion, we now consider an active user who

inspects products in the ordering $\nu = (1, \ldots, N)$, rating each based on the prediction of the NN algorithm. It is easy to see that when there are no manipulators, the NN algorithm perfectly predicts $\hat{x}_{k,x^{k-1},Y} = \mathbf{x}_k^{\text{odd}}$, and therefore, after the user inspects k products, his ratings history x^k has $x_j^k = \mathbf{x}_j^{\text{odd}}$ for $j \leq k$ and $x_j^k = \circ$ for $j > k$.

We assume that manipulators produce one half of the training data. For each honest ratings vector y^m, manipulators produce a ratings vector z^m which agrees with y^m on all products rated by y^m. However, circles in y^m are replaced by 1 for even indices and 0 for odd indices. That is, each z_m corresponds to an element of the set $\{(1,0),(0,1)\}^{N/2}$.

Suppose k is even. Given x^k, the NN algorithm predicts what the active user's rating will be for product $k+1$. To do this, it identifies neighbors $\mathcal{N}(k+1, x^k, (Y,Z))$, which includes the following subsets of the training data:

- A set Y_1 which consists of honest ratings vectors y^m where $y_j^m \neq \circ$ for $j \leq k+1$.
- A set Z_1 that, for each $y^m \in Y_1$, includes the corresponding manipulated vector $z^m \in Z$.
- A set Z_2 which consists of each manipulated ratings vector z^m such that the corresponding honest ratings vector y^m has $y_j^m \neq \circ$ for $j \leq k$ and $y_{k+1}^m = \circ$.

Note that each of these sets is of cardinality $2^{(N-k)/2-1}$. Vectors in Y_1 and Z_1 correctly rate product $k+1$ as 1, whereas vectors in Z_2 incorrectly rate it as 0. As a consequence, the prediction for product $k+1$ is $\hat{x}_{k+1,x^k,(Y,Z)} = 2/3$ and the resulting squared error is

$$\left(\mathbf{x}_{k+1}^{\text{odd}} - \hat{x}_{k+1,x^k,(Y,Z)}\right)^2 = \frac{1}{9}.$$

The preceding argument applies for all even k. For odd k, it is easy to show that the NN algorithm correctly predicts $\mathbf{x}_{k+1}^{\text{odd}} = 0$. It follows that the RMS distortion for even n is

$$d_n^{\text{RMS}}(p,\nu,Y,Z) = \sqrt{\frac{1}{n}\sum_{k=1}^{n} \mathrm{E}\left[\left(\mathbf{x}_k^{\text{odd}} - \hat{x}_{k,x^{k-1},(Y,Z)}\right)^2\right]} = \frac{1}{3\sqrt{2}}.$$

The preceding example shows that the RMS distortion of an NN algorithm for $r = 1/2$ does not decrease as n grows. This happens because manipulated data are strategically generated to be sufficiently similar to honest data so that no matter how many ratings an active user provides, manipulated ratings vectors will make up a fixed fraction of the neighbors and consequently induce a significant amount of distortion.

In contrast, Corollary 1 establishes that linear CF algorithms exhibit a more graceful behavior, with RMS distortion vanishing as n increases. This is not to say it is impossible to design an NN algorithm that exhibits a more desirable behavior when applied to our example. However, it is difficult to know for sure whether a given variation will behave gracefully in all relevant situations.

4.5 Discussion

We now provide an intuitive explanation for why linear CF algorithms should be robust to manipulation relative to NN algorithms. First note that robustness depends on how a CF algorithm learns from its mistakes. In particular, a robust algorithm should notice as it observes differences between its predictions and an active user's ratings that certain things learned from the data set are hurting rather than improving its predictions.

Recall that a linear CF algorithm P generates based on the training set (Y, Z) a PMF $P(\cdot \mid (Y, Z))$ that is a convex combination of $P(\cdot \mid Y)$ and $P(\cdot \mid Z)$, which are PMFs that the algorithm would generate based on Y and Z, respectively. As an active user rates more products, it will be increasingly clear by probabilistic inference that his ratings x are sampled from $P(\cdot \mid Y)$. In effect, inaccurate predictions induced by Z will increase the weight on $P(\cdot \mid Y)$ in the conditional PMF of ratings $P(\cdot \mid x, (Y, Z))$ conditioned on observed ratings x. And this makes future predictions more accurate.

In an NN algorithm, on the other hand, inaccurate predictions do not generally improve further predictions. In particular, manipulated ratings vectors that contribute to inaccuracies may remain in the set of neighbors while honest ratings vectors may be eliminated from it. In the example in Section 4.4, for instance, manipulated data are generated so that no matter how long an active user's ratings history is, each honest ratings vector selected as a neighbor has a manipulated counterpart that is as similar, and hence also selected as a neighbor. Consequently, as an active user provides more ratings, the numbers of honest and manipulated neighbors both decrease and stay equal. As a result, inaccurate predictions do not decrease future distortion.

5 Empirical Study

In this section, we present our empirical findings on the manipulation robustness of NN, linear, and asymptotically linear CF algorithms. We first introduce the data set that we worked with and then describe the methods we used to evaluate robustness.

5.1 Data Set

We obtained a set of movie ratings provided by users, made publicly available by Netflix's recommendation system. Each rating is an integer between 1 and 5, which we normalized to be in $\{0, 0.25, 0.5, 0.75, 1\}$ so that the analysis and results in our paper apply directly. We randomly sampled from the data set 5000 users and 500 movies. The number of ratings provided by these users on these movies is around 200000. We then randomly chose 4000 of these users and for the purpose of our experiments, treated them as honest users and their ratings as a training set Y. We used the ratings of the other 1000 users as a test set, which we refer to as X. We then generated three separate sets of 444, 1714, and 4000 manipulated ratings vectors, respectively. Each set,

which we refer to as Z for simplicity of discussion, is generated to promote 50% of the movies by using a technique reported to be effective in the literature (Burke et al., 2005; Lam and Riedl, 2004; Mehta and Nejdl, 2008; Mobasher et al., 2005, 2006; O'Mahony et al., 2004; Sandvig et al., 2007; Zhang et al., 2006). Specifically, we randomly sampled 250 of the 500 movies in Y and let each manipulated ratings vector in each Z assign the highest ratings to these movies, and assign a random rating to each of the other movies, sampled from the movie's empirical marginal PMF of ratings in Y. We then replaced a random subset of ratings in Z with circles so that its fraction of circles matches that in Y. Manipulated ratings vectors generated this way are meant to be similar to honest ratings vectors except on movies to promote.

5.2 Evaluation Methods

To test the robustness of each CF algorithm P, we treated ratings in X as ratings that an active user would provide and let P predict them. Specifically, we fixed n and for each ratings vector $x \in X$, identified n random products that it has assigned ratings to and randomly permuted them to form an ordering $\nu^x = (\nu_1^x, \ldots, \nu_n^x)$. For each $k \leq n$, let x^{k-1} be a ratings vector that agrees with x on products $\nu_1^x, \ldots, \nu_{k-1}^x$ and assigns circles to the other products. An algorithm P is then used to generate a scalar prediction $\hat{x}_{\nu_k^x, x^{k-1}, Y}$ for the rating of product ν_k^x based on x^{k-1} and the honest data set Y. Similarly, a prediction based on a training set (Y, Z) corrupted by manipulated ratings is denoted by $\hat{x}_{\nu_k^x, x^{k-1}, (Y,Z)}$. To assess influence due to manipulation, for each n, we computed the following quantity, which we will refer to as *empirical RMS distortion*:

$$\hat{d}_n^{\text{RMS}}(P, \nu^X, X, Y, Z) = \sqrt{\frac{1}{|X|} \sum_{x \in X} \frac{1}{n} \sum_{k=1}^{n} \left(\hat{x}_{\nu_k^x, x^{k-1}, Y} - \hat{x}_{\nu_k^x, x^{k-1}, (Y,Z)} \right)^2}.$$

Here, $\nu^X = \{(\nu_1^x, \ldots, \nu_n^x) : x \in X\}$. The empirical RMS distortion measures changes of predictions for products rated by active users. It is similar to the RMS distortion $d_n^{\text{RMS}}(P, \nu, Y, Z)$ that we defined earlier, with one difference: whereas $d_n^{\text{RMS}}(P, \nu, Y, Z)$ samples each $\mathbf{x}_{\nu_k}$ from the PMF $P\left(\cdot \mid x^{k-1}, Y, \nu\right)$ that the algorithm generates based on Y, $\hat{d}_n^{\text{RMS}}(P, \nu^X, X, Y, Z)$ uses elements of X as samples. We used empirical RMS distortion rather than RMS distortion to assess algorithms in our empirical study because computing RMS distortion would take too long, requiring a running time exponential in the number of products n rated by an active user. Further, if a CF algorithm generates a nearly correct distribution in the absence of manipulation, its empirical RMS distortion will be close to its RMS distortion.

One might also wonder whether high robustness of CF algorithms stems from high prediction accuracy or comes at the expense of it. To better understand the relationship between these two performance measures, we also computed the following RMS error for each CF algorithm, which

we will refer to as *empirical RMS prediction error*:

$$\hat{\mathcal{E}}_n^{\text{RMS}}(P, \nu^X, X, Y) = \sqrt{\frac{1}{|X|} \sum_{x \in X} \frac{1}{n} \sum_{k=1}^{n} \left(x_{\nu_k^x} - \hat{x}_{\nu_k^x, x^{k-1}, Y} \right)^2}.$$

This quantity computes the RMS error of predictions for ratings in X when the algorithm uses Y as training data.

For algorithms that we tested, we tuned some of their parameters by cross validation. This is a technique that selects parameter values based on the performance of the corresponding algorithm on out-of-sample data in order to estimate their performance on future data. Specifically, we randomly sampled 20% of the users in Y. We treated their ratings as a validation set V and generated predictions based on the remaining ratings $Y \backslash V$. Consider a parameter γ that we tuned for an algorithm. For each value γ' in a range Γ, we set $\gamma = \gamma'$, used the corresponding algorithm to predict ratings in V based on ratings in $Y \backslash V$, and computed the empirical RMS prediction error $\hat{\mathcal{E}}_n^{\text{RMS}}(P, \nu^V, V, Y \backslash V)$. Finally, we selected a parameter value γ^* that results in a minimal error. Similarly, when using (Y, Z) as the training set, we sampled the validation set V from (Y, Z) and for each $\gamma' \in \Gamma$, computed the empirical RMS prediction error $\hat{\mathcal{E}}_n^{\text{RMS}}(P, \nu^V, V, (Y, Z) \backslash V)$ and selected a γ^*. Note that we chose to optimize for prediction accuracy rather than robustness in cross validation because we wanted the algorithms to maintain reasonable accuracy and wanted to avoid tuning them to be robust for specific manipulation techniques.

Overall, for each algorithm P, we generated multiple samples of X, Y, Z, and ν^X, and averaged their resultant $\hat{d}_n^{\text{RMS}}(P, \nu^X, X, Y, Z)$ and $\hat{\mathcal{E}}_n^{\text{RMS}}(P, \nu^X, X, Y)$ across samples to obtain reliable estimates. To summarize with our notation, $\mathbf{S} = \{0, 0.25, 0.5, 0.75, 1\}$, $N = 500$, $(M, r) \in \{(4444, 0.1), (5714, 0.3), (8000, 0.5)\}$, and $1 \leq n \leq 40$. We tested three CF algorithms: a linear CF algorithm called kernel density estimation, an asymptotically linear CF algorithm called naive Bayes, and an NN algorithm called k nearest neighbor. We now present them in detail.

5.3 Kernel Density Estimation Algorithms

Kernel density estimation (KDE) algorithms smooth the training data and use their resultant distribution to predict future ratings. For an in-depth treatment of KDE algorithms, see Hastie et al. (2001). In our context, we say that a probabilistic CF algorithm P is a KDE algorithm with kernels $\{\mathcal{K}_w : w \in S^N\}$ if for any $W \in S^{N \times M}$,

$$P(\cdot \mid W) = \frac{1}{M} \sum_{w \in W} \mathcal{K}_w(\cdot),$$

where each $\mathcal{K}_w$ is a PMF over $\mathbf{S}^N$ parameterized by a ratings vector w. It turns out that any KDE algorithm is a linear CF algorithm and any linear CF algorithm is a KDE algorithm. We will establish this in Proposition 3 in Appendix A.3.

In our experiments, we considered a KDE algorithm with kernels $\{\mathcal{K}_w\}$ such that for each type $\mathbf{x} \in \mathbf{S}^N$,

$$\mathcal{K}_w(\mathbf{x}) = \prod_{n=1}^{N} k_{w_n}(\mathbf{x}_n),$$

where for each $s \in S$, k_s is a PMF over $\mathbf{S}$ defined as follows. For $s \neq \circ$, k_s is the unique PMF that satisfies $k_s(\mathbf{s})/k_s(\mathbf{s}) = \exp(-|\mathbf{s} - s|/\beta)$ for all $\mathbf{s} \in \mathbf{S}$. For $s = \circ$, $k_s(\mathbf{s}) = 1/|\mathbf{S}|$ for all $\mathbf{s} \in \mathbf{S}$. That is, k_s assigns the highest probability to s and exponentially lower probabilities to values different from s if $s \neq \circ$, and assigns uniform probability to all values if $s = \circ$. It is easy to see that each $\mathcal{K}_w$ thus defined is a PMF, and it assigns high probability to types similar to w and low probability to others. The constant $\beta > 0$ tunes the shape of k_s, which we set to be 0.15 in our experiments.

To predict the rating of product ν_n for a user with past ratings x^{n-1}, our KDE algorithm generates a PMF given by

$$P\left(\mathbf{x}_{\nu_n} = \mathbf{s} \,\middle|\, x^{n-1}, W\right) = \frac{\sum_{w \in W} \prod_{i=1}^{n-1} k_{w_{\nu_i}}(x_{\nu_i}^{n-1}) k_{w_{\nu n}}(\mathbf{s})}{\sum_{w \in W} \prod_{i=1}^{n-1} k_{w_{\nu_i}}(x_{\nu_i}^{n-1})},$$

for each $\mathbf{s} \in \mathbf{S}$. The corresponding scalar prediction is the expectation taken with respect to $P(\cdot \mid x_{n-1}, W)$:

$$\hat{x}_{\nu_n, x^{n-1}, W} = \sum_{\mathbf{s} \in \mathbf{S}} \mathbf{s}\, P\left(\mathbf{x}_{\nu_n} = \mathbf{s} \,\middle|\, x^{n-1}, W\right).$$

5.4 Naive Bayes Algorithm

A naive Bayes (NB) algorithm assumes that the true distribution of data is a convex combination of distinct distributions in each of which features of the data are conditionally independent. It aims to learn from training data the weights of the combination and feature marginals within each distribution. For a formal analysis of the algorithm and its applications to other problem settings, see Cheeseman and Stutz (1996); Domingos and Pazzani (1997); John and Langley (1995). We now describe a particular version of the algorithm that we used and discuss the context in which it is consistent and asymptotically linear.

Our NB algorithm assumes that data are generated in the following way. First, user types are sampled from a distribution P^* where

$$P^*(\mathbf{w}) = \sum_{l=1}^{L} \eta_l \prod_{n=1}^{N} \theta_{l,n}^{(\mathbf{w}_n)}, \tag{2}$$

and ratings vectors are sampled from the conditional distribution Q^* where

$$Q^*(w \mid \mathbf{w}) = q^{\|w\|_\circ}(1-q)^{N-\|w\|_\circ}, \text{ if } \mathbf{w} \to w, \text{ and} \tag{3}$$

$$Q^*(w \mid \mathbf{w}) = 0, \text{ otherwise.}$$

Here, $q \in [0,1)$ and $L \in \mathcal{Z}_+$. $\eta \in \Delta_L$ and $\theta_{l,n} \in \Delta_{|\mathbf{S}|}$ for all l, n, where $\Delta_k = \{(d_1, \ldots, d_k) : \forall\, i, d_i \geq 0, \sum_j d_j = 1\}$ denotes a simplex. We write $\mathbf{w} \to w$ for $(\mathbf{w}, w)$ if for each n, either $w_n = \mathbf{w}_n$ or $w_n = \circ$. We let $\|w\|_\circ$ denote $|\{n : w_n = \circ\}|$ and let $\theta = \{\theta_{l,n}, 1 \leq l \leq L, 1 \leq n \leq N\}$.

To understand P^* and Q^*, let us consider the following generative process of ratings vectors. Let $\mathcal{P} = \{P_1^*, \ldots, P_L^*\}$ be L PMFs over $\mathbf{S}^N$ where each P_l^* satisfies

$$P_l^*(\mathbf{w}) = \prod_{n=1}^{N} \theta_{l,n}^{(\mathbf{w}_n)}$$

for all $\mathbf{w} \in \mathbf{S}^N$. That is, each $\mathbf{w}_n$ is independently distributed and is equal to $\mathbf{s} \in \mathbf{S}$ with probability $\theta_{l,n}^{(\mathbf{s})}$. A type $\mathbf{w}$ is generated by first selecting a PMF from $\mathcal{P}$ where each P_l^* is chosen with probability η_l and then sampling from that PMF. A ratings vector w is then generated by randomly replacing each rating $\mathbf{w}_n$ by a circle with probability q, independent of the value $\mathbf{w}_n$ and whether other ratings are replaced by circles.

The algorithm also assumes a geometric prior for L and Dirichlet priors for η, θ, and q. Hence, the posterior probability density function (PDF) of (L, η, θ, q) conditioned on training data W is given by

$$f(L, \eta, \theta, q|W) = c f(L, \eta, \theta, q) \Pr(W|L, \eta, \theta, q), \tag{4}$$

where c is a normalizing constant and prior PDF

$$f(L, \eta, \theta, q) = f_L^\tau(L) f_\eta(\eta) \prod_{l,n} f_\theta(\theta_{l,n}) f_q(q), \text{ with}$$

$$f_L^\tau(L) = c_L e^{-\tau L},$$

$$f_\eta(\eta) = c_\eta \prod_{l=1}^{L} \eta_l,$$

$$f_\theta(\theta_{l,n}) = c_\theta \prod_{\mathbf{s} \in \mathbf{S}} \theta_{l,n}^{(\mathbf{s})}, \forall\, l, n, \text{ and}$$

$$f_q(q) = c_q q(1-q),$$

and data likelihood

$$\Pr(W|L, \eta, \theta, q) = \prod_{w \in W} \left(\left(q^{\|w\|_\circ} (1-q)^{N - \|w\|_\circ} \right) \left(\sum_{l=1}^{L} \eta_l \prod_{n : w_n \neq \circ} \theta_{l,n}^{(w_n)} \right) \right).$$

Here, subscripts L, η, θ, and q of the functions f_L^τ, f_η, f_θ, and f_q denote the parameters that the distributions are over. c_L, c_η, c_θ, and c_q are normalizing constants.

The algorithm maximizes the posterior PDF over parameters by using the expectation-maximization

algorithm (Dempster et al., 1977) and obtains

$$(\hat{L}, \hat{\eta}, \hat{\theta}, \hat{q}) \in \operatorname*{argmax}_{(L,\eta,\theta,q)} f(L, \eta, \theta, q \mid W).$$

We denote by $P(\cdot \mid W)$ the PMF over $\mathbf{S}^N$ implied by $(\hat{L}, \hat{\eta}, \hat{\theta})$, which the algorithm uses for predictions.

In particular, a prediction for the rating of product ν_n for a user with past ratings x^{n-1} is given by the PMF

$$P\left(\mathbf{x}_{\nu_n} = \mathbf{s} \;\middle|\; x^{n-1}, W\right) = \frac{\sum_{l=1}^{\hat{L}} \hat{\eta}_l \prod_{k=1}^{n-1} \hat{\theta}_{l,\nu_k}^{(x_{\nu_k}^{n-1})} \hat{\theta}_{l,\nu_n}^{(\mathbf{s})}}{\sum_{l=1}^{\hat{L}} \hat{\eta}_l \prod_{k=1}^{n-1} \hat{\theta}_{l,\nu_k}^{(x_{\nu_k}^{n-1})}},$$

for each $\mathbf{s} \in \mathbf{S}$. The corresponding scalar prediction is

$$\hat{x}_{\nu_n, x^{n-1}, W} = \sum_{\mathbf{s} \in \mathbf{S}} \mathbf{s}\, P\left(\mathbf{x}_{\nu_n} = \mathbf{s} \;\middle|\; x^{n-1}, W\right).$$

In our experiments, we tuned bandwidth τ by cross validation over the range $\Gamma = \{1, 10, 100, 1000, 10000, 100000\}$ and settled at $\tau = 10000$.

We now discuss the context in which the NB algorithm is consistent and asymptotically linear. For any $q \in [0, 1)$, we let Φ_q be the set of all $(\overline{P}, \overline{Q}_q)$ pairs where $\overline{P}$ takes the form in (2) and $\overline{Q}_q$ is the one distribution given by (3), with q fixed. We establish in Proposition 4 in Appendix A.3 that for any q, Φ_q is identifiable and convex, and the NB algorithm is consistent with respect to it. Then, by Theorems 2 and 3, the algorithm is asymptotically linear with respect to Φ_q and our distortion bounds apply. Note that the set $\mathcal{P} = \{\overline{P} : (\overline{P}, \overline{Q}_q) \in \Phi_q\}$ of type PMFs with corresponding pairs in $\mathcal{P}_q$ is the set of all PMFs over $\mathbf{S}^N$. This implies that for any PMFs $\overline{P}_1$ and $\overline{P}_2$ over $\mathbf{S}^N$, if honest and manipulated data are generated by first sampling types from these PMFs and then independently replacing each rating by a circle with the same probability q, then the NB algorithm will be asymptotically robust as the sample size grows. In our experiments, although the condition regarding replacement by circles may not hold, we still apply the NB algorithm with the hope that it will deliver reasonable robustness.

5.5 k Nearest Neighbor Algorithm

A class of NN algorithms called k nearest neighbor (kNN) algorithms is frequently used as a performance benchmark in prior work (Burke et al., 2005; Lam and Riedl, 2004; Zhang et al., 2006). The version that we tested works as follows.

To predict the rating of product ν_n by a user with past ratings x^{n-1} where $n \geq 3$, the algorithm identifies a set of neighbors $\mathcal{N}(\nu_n, x^{n-1}, W)$ to be k ratings vectors $w \in W$ such that $w_{\nu_n} \neq \circ$ and score highest with x^{n-1} on the following similarity measure, which is sometimes referred to as the

cosine similarity measure in the literature Sarwar et al. (2001):

$$s(w, x^{n-1}) = \frac{\sum_{1 \le i \le n-1 : w_{\nu_i} \neq \circ} (w_{\nu_i} - \dot{w})(x^{n-1}_{\nu_i} - \dot{x}^{n-1})}{\sqrt{\sum_{1 \le i \le N : w_i \neq \circ} (w_i - \dot{w})^2} \sqrt{\sum_{1 \le i \le n-1} (x^{n-1}_{\nu_i} - \dot{x}^{n-1})^2}},$$

where average ratings are given by

$$\dot{w} = \frac{\sum_{1 \le i \le N : w_i \neq \circ} w_i}{|\{1 \le i \le N : w_i \neq \circ\}|},$$

$$\dot{x}^{n-1} = \frac{\sum_{1 \le i \le n-1} x^{n-1}_{\nu_i}}{n-1}.$$

Note that s here resembles the notion of a sample correlation coefficient. Its numerator is proportional to the covariance between non-question-mark components of w and x^{n-1}. The denominator is proportional to the product of the standard deviation of non-question-mark components of w and the same quantity for x^{n-1}. The algorithm then generates the following scalar prediction:

$$\hat{x}_{\nu_n, x^{n-1}, W} = \min \left\{ \mathbf{s}_{\max}, \max \left\{ \mathbf{s}_{\min}, \dot{x}^{n-1} + \frac{\sum_{w \in \mathcal{N}(\nu_n, x^{n-1}, W)} s(w, x^{n-1})(w_{\nu_n} - \dot{w})}{\sum_{w \in \mathcal{N}(\nu_n, x^{n-1}, W)} |s(w, x^{n-1})|} \right\} \right\},$$

where $\mathbf{s}_{\max} = \max\{\mathbf{s} : \mathbf{s} \in \mathbf{S}\}$ and $\mathbf{s}_{\min} = \min\{\mathbf{s} : \mathbf{s} \in \mathbf{S}\}$. To arrive at this quantity, for each neighbor, the difference between its rating w_{ν_n} for product ν_n and its average rating $\dot{w}$ is first computed. A weighted sum of these differences is then computed, where the weights are normalized similarity measures. The user's historical ratings average $\dot{x}^{n-1}$ is then added to the sum. The total is used as the prediction, unless it falls outside $[\mathbf{s}_{\min}, \mathbf{s}_{\max}]$, in which case either $\mathbf{s}_{\min}$ or $\mathbf{s}_{\max}$ is used, whichever is closer.

For a user with ratings history x^{n-1} where $n \le 2$, $s(\cdot, x^{n-1})$ is not well-defined. In this case, the algorithm uses the average rating of product ν_n in the training data to generate the prediction:

$$\hat{x}_{\nu_n, x^{n-1}, W} = \min \left\{ \mathbf{s}_{\max}, \max \left\{ \mathbf{s}_{\min}, \frac{\sum_{\{w : w \in W, w_{\nu_n} \neq \circ\}} w_{\nu_n}}{|\{w : w \in W, w_{\nu_n} \neq \circ\}|} \right\} \right\}.$$

In our experiments, we tuned the number of neighbors k by cross validation over the range $\Gamma = \{1, 2, \ldots, 40\}$ and settled at $k = 10$.

Note that even though the kNN algorithm generates scalar predictions, it still fits our definition of CF algorithms because it is possible to come up with PMFs whose corresponding expectations equal the predictions $\hat{x}_{\nu_n, x^{n-1}, W}$. We do not explicitly define such a PMF, however, because it is not necessary for computing the empirical RMS distortion in our experiments.

5.6 Results

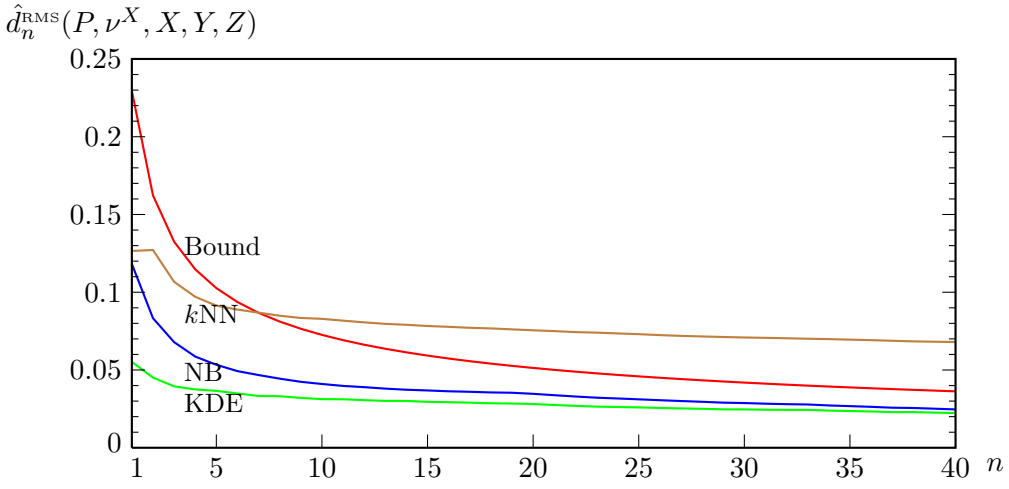
Figure 2 shows the empirical RMS distortions for the three algorithms that we tested, with different fractions of manipulated data. Our results suggest that in practice, NB and KDE algorithms are significantly more robust than kNN. In particular, when a user's ratings history is short, kNN and NB both incur higher empirical RMS distortions than KDE. This difference arises because while kNN and NB ignore circles, KDE uses them and as a result, tempers its predictions. To gain some intuition, let us consider the following problem instance where ratings are binary: the set of honest ratings Y consists of K vectors whose entries are all 1s and as many vectors whose entries are all circles. The set of manipulated ratings Z consists of K vectors whose entries are all 0s and as many vectors whose entries are all circles. To predict the first rating $\mathbf{x}_{\nu_1}$ of an active user, kNN would yield a prediction of 1 and 1/2 based on Y and (Y, Z), respectively, incurring an RMS distortion of 1/2. KDE would yield a prediction close to 3/4 based on Y and a prediction of 1/2 based on (Y, Z), incurring an RMS distortion of 1/4, significantly less than that of kNN. Clearly, the presence of circles smooths KDE's predictions and keeps its distortion low.

In Figure 2, as more ratings are provided, distortions incurred by all three algorithms decrease. When a user's ratings history is long, NB and KDE incur distortions significantly lower than that of kNN. Note that distortions of NB and KDE always stay below the bound in Corollary 1. The curves for kNN are flat for $n \leq 2$ because the algorithm provides the same predictions for the first two ratings $\mathbf{x}_{\nu_1}$ and $\mathbf{x}_{\nu_2}$ of an active user. Note that as fraction of manipulated data r increases, distortions incurred by all three algorithms increase as well.

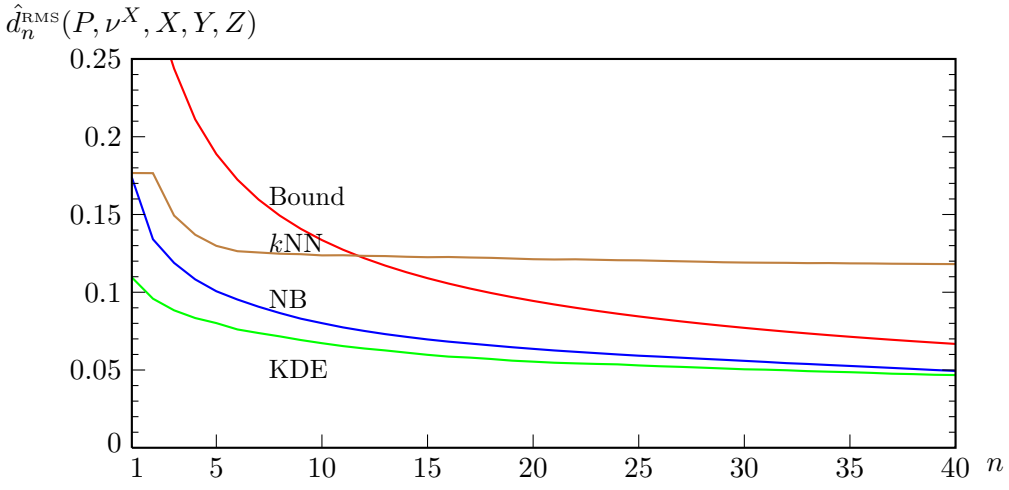
Figure 3 displays the empirical RMS prediction errors of the three algorithms. When n is large, their errors all decrease and in particular, NB offers the lowest error and kNN, the highest. kNN sees a spike around $n = 3$ because the algorithm switches its prediction method there: it generates predictions by using average ratings of all users for $n \leq 2$ and generates predictions by using average ratings of neighbors for $n \geq 3$.

To get a better sense of our results, we note that Netflix announced that its proprietary algorithm achieves an empirical RMS prediction error, normalized to our scale, of 0.238 on a large test set, and will award one million dollars to anyone that improves it to 0.214 (Netflix Prize, 2006). One might wonder why a decrease of 0.024 may have such a large impact on recommendation quality. We suspect that due to the large number of movies, many of them are given similar predicted ratings. As a result, a small improvement in prediction accuracy may tease apart these movies and identify the most desirable ones.

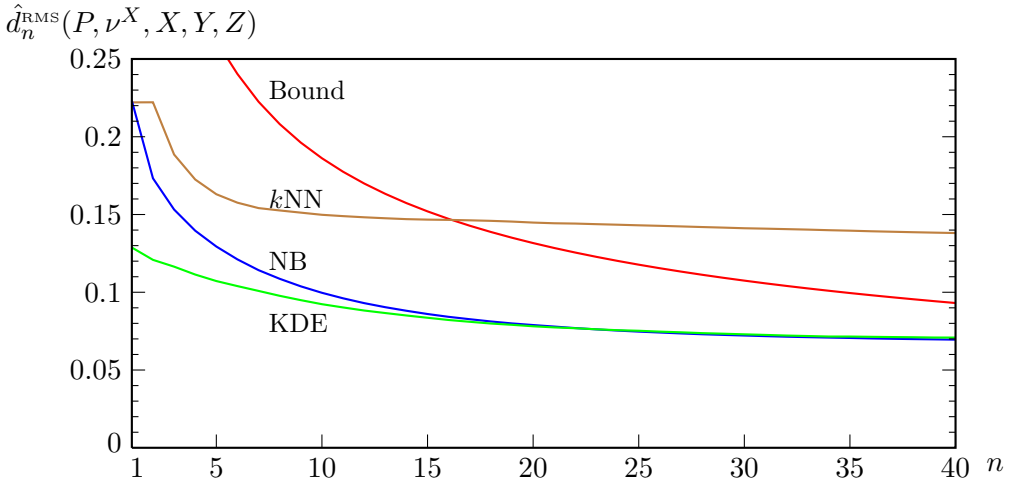
Compared to Netflix's benchmark and target prediction errors, our results are reasonable but not competitive. This is because we did not focus on optimizing the prediction accuracy of the algorithms. If our objective was to achieve the highest possible accuracy while maintaining reasonable robustness, one option we could try is to fine-tune our robust algorithms to be accurate. For example, for KDE algorithms, we could work to identify more effective kernels. For NB algorithms, we could choose different priors or use methods other than expectation-maximization to find the



(a) $r = 0.1$



(b) $r = 0.3$



(c) $r = 0.5$

Figure 2: Empirical RMS distortion as a function of n, for different r.

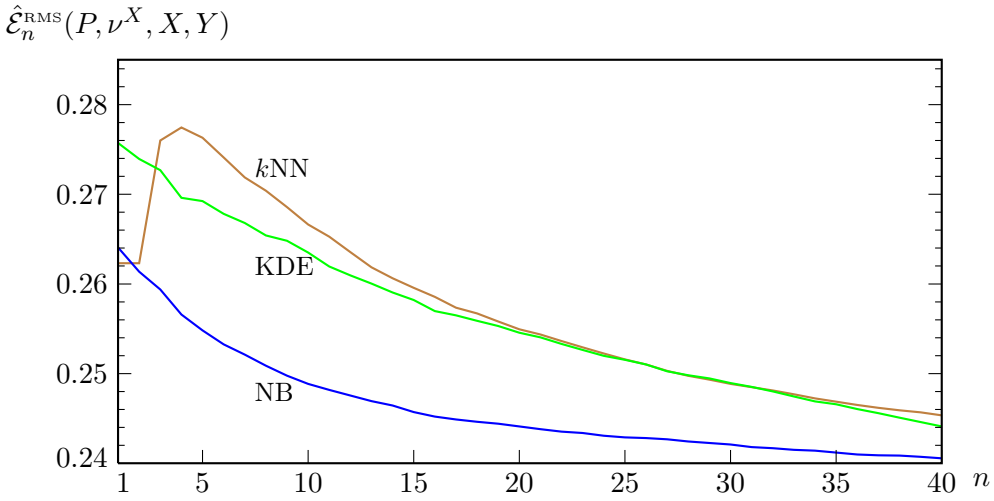


Figure 3: Empirical RMS prediction error as a function of n.

model parameters. We could probably also design other robust linear and asymptotically linear CF algorithms that achieve higher accuracy as well. Overall, we are not suggesting that in practice, the specific algorithms that we presented should be directly implemented. Instead, one should either use them as starting points or take the insights that they yield into consideration when designing accurate and robust CF systems.

6 Extensions

Our analytical and empirical work suggests that linear and asymptotically linear algorithms can be more robust to manipulation than commonly used nearest neighbor algorithms. Our results also suggest that it is possible to design algorithms that achieve accuracy alongside robustness. As such, recommendation systems of Internet commerce sites may improve their robustness to manipulation by adopting the approaches that we describe. They may also use the bounds on distortion that we establish as a guide on how many ratings each user should provide to a recommendation system before its predictions can be trusted.

The simple setting in our work serves as a context for the initial development of our idea, and can be extended in multiple ways. One direction is to study the robustness of collaborative filtering algorithms as measured by alternative metrics. One metric could be, for instance, a user's utility loss due to manipulation. Another extension is to design algorithms that provide non-asymptotic guarantees on both prediction accuracy and robustness.

The framework that we establish also facilitates studying the effectiveness of alternative techniques to abate influence by manipulators. For instance, given a scheme that incentivizes users to inspect and rate products, one could analyze how honest users and manipulators would behave, and then use our distortion metrics to assess the robustness of the scheme to manipulation.

It is also worth mentioning that many commercial recommendation systems build on multiple sources of information, not just collaborative filtering (Adomavicius and Tuzhilin, 2005). For example, as discussed in Balabanovic and Shoham (1997), recommendations should also be guided by features of the products being recommended. Our preliminary investigation suggests that the robustness of collaborative filtering algorithms that make use of product features parallels that of the algorithms that we discussed in this paper. For example, there exist "linear" feature-guided collaborative filtering algorithms to which our distortion bounds apply.

Acknowledgments

The authors thank Christina Aperjis, Thomas Cover, Paul Cuff, Amir Dembo, Persi Diaconis, Vivek Farias, John Gill, Ramesh Johari, Yi-hao Kao, Yi Lu, Taesup Moon, Beomsoo Park, and Assaf Zeevi for helpful suggestions. This research was supported in part by the NET Institute (www.netinst.org) and by the National Science Foundation through grant IIS-0428868.

References

Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

Marko Balabanovic and Yoav Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

James Bennett. The Cinematch system: operation, scale coverage, accuracy impact. http://blog.recommenders06.com/wp-content/uploads/2006/09/1jimbennett.wmv, 2006.

Rajat Bhattacharjee and Ashish Goel. Algorithms and incentives for robust ranking. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2007.

John Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, 1998.

Robin Burke, Bamshad Mobasher, Roman Zabicki, and Runa Bhaumik. Limited knowledge shilling attacks in collaborative filtering systems. In *Proceedings of the Third IJCAI Workshop in Intelligent Techniques for Personalization*, 2005.

Peter Cheeseman and John Stutz. Bayesian classification (AutoClass): Theory and results. In Usama Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI, 1996.

Chrysanthos Dellarocas. Strategic manipulation of internet opinion forums: Implications for consumers and firms. *Management Science*, 52(10):1577–1593, 2006.

Arthur Dempster, Nan Laird, and Donald Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1):1–38, 1977.

Pedro Domingos and Michael Pazzani. On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29(2):103–130, 1997.

Petros Drineas, Iordanis Kerenidis, and Prabhakar Raghavan. Competitive recommendation systems. In *Proceedings of the Thirty-fourth Annual ACM Symposium on Theory of Computing*. ACM, 2002.

Eric Friedman, Paul Resnick, and Rahul Sami. Manipulation-resistant reputation systems. In Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay Vazirani, editors, *Algorithmic Game Theory*, chapter 27, pages 677–697. Cambridge University Press, 2007.

Olivier Gossner and Tristan Tomala. Entropy bounds on Bayesian learning. *Journal of Mathematical Economics*, 44:24–32, 2008.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2001.

Jonathan Herlocker, Joseph Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the Twenty-second Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 1999.

George John and Pat Langley. Estimating continuous distributions in Bayesian classifiers. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 1995.

John Kleinberg and Mark Sandler. Using mixture models for collaborative filtering. *Computer and System Sciences*, 74(1):49–69, 2008.

Shyong Lam and John Riedl. Shilling recommender systems for fun and profit. In *Proceedings of the Thirteenth International Conference on World Wide Web*. ACM, 2004.

Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

Paolo Massa and Paolo Avesani. Trust-aware collaborative filtering for recommender systems. In *Proceedings of the Eleventh International Conference on Intelligent User Interfaces*. ACM, 2008.

Bhaskar Mehta. Unsupervised shilling detection for collaborative filtering. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence*. AAAI, 2007.

Bhaskar Mehta and Wolfgang Nejdl. Attack resistant collaborative filtering. In *Proceedings of the Thirty-First Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2008.

Nolan Miller, Paul Resnick, and Richard Zeckhauser. Eliciting informative feedback: The peer-prediction method. *Management Science*, 51(9):1359–1373, 2005.

Bamshad Mobasher, Robin Burke, Runa Bhaumik, and Chad Williams. Effective attack models for shilling item-based collaborative filtering systems. In *Proceedings of the 2005 WebKDD Workshop*. ACM, 2005.

Bamshad Mobasher, Robin Burke, and JJ Sandvig. Model-based collaborative filtering as a defense against profile injection attacks. In *Proceedings of the Twenty-first National Conference on Artificial Intelligence*. AAAI, 2006.

Bamshad Mobasher, Robin Burke, Runa Bhaumik, and Chad Williams. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *ACM Transactions on Internet Technology*, 7(4), 2007.

Sangkil Moon and Gary Russell. Predicting product purchase from inferred customer similarity: an autologistic model approach. *Management Science*, 54(1):71–82, 2008.

Rajeev Motwani and Sergei Vassilvitskii. Tracing the path: new model and algorithms for collaborative filtering. In *Twenty-third International Conference on Data Engineering Workshop*. IEEE, 2007.

Netflix Prize. http://www.netflixprize.com, 2006.

John O'Donovan and Barry Smyth. Is trust robust? An analysis of trust-based recommendation. In *Proceedings of the Eleventh International Conference on Intelligent User Interfaces*. ACM, 2006.

Stefanie Olsen. Amazon blushes over sex link gaffe. http://news.cnet.com/2100-1023-976435.html, 2002.

Michael O'Mahony, Neil Hurley, Nicholas Kushmerick, and Guenole Silvestre. Collaborative recommendation: A robustness analysis. *ACM Transactions on Internet Technologies*, 4(4):344–377, 2004.

Paul Resnick and Rahul Sami. The Influence Limiter: provably manipulation-resistant recommender systems. In *Proceedings of the ACM Recommender Systems Conference*. ACM, 2007.

Lucas Ryan. Personal communication, 2008.

JJ Sandvig, Bamshad Mobasher, and Robin Burke. Robustness of collaborative recommendation based on association rule mining. In *Proceedings of the 2007 ACM Conference on Recommender Systems*. ACM, 2007.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the Tenth International Conference on World Wide Web*. ACM, 2001.

J. Ben Schafer, Joseph Konstan, and John Riedl. E-commerce recommendation applications. *Data Mining and Knowledge Discovery*, 5(1):115–153, 2001.

Abraham Wald. Note on the consistency of the maximum likelihood estimate. *The Annals of Mathematical Statistics*, 20(4):595–601, 1949.

Chad Williams, Bamshad Mobasher, and Robin Burke. Defending recommender systems: Detection of profile injection attacks. *Journal of Service Oriented Computing and Applications*, 1(3), 2007.

Sheng Zhang, Yi Ouyang, James Ford, and Fillia Makedon. Analysis of a low-dimensional linear model under recommendation attacks. In *Proceedings of the Twenty-Ninth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2006.

A Proofs

A.1 Relationships Among Distortion Measures

Propositions 1 and 2 state relationships between KL, RMS, and binary prediction distortions. Lemmas 1 and 2 help prove them.

Lemma 1. *Let u and v be random variables distributed according to PMFs p and q, both with support on the same finite set $U \subset [0, 1]$. It holds that*

$$|\mathrm{E}[u] - \mathrm{E}[v]| \leq \frac{1}{2}\|p - q\|_1.$$

Proof. Let $U = \{u_1, \ldots, u_N\}$ and correspondingly, let $p_i = p(u_i)$ and $q_i = q(u_i)$, for $1 \leq i \leq N$. Without loss of generality, let $p_1 - q_1 \geq p_2 - q_2 \geq \cdots \geq p_N - q_N$. There exists n such that $p_n - q_n \geq 0 \geq p_{n+1} - q_{n+1}$. Hence, $\sum_{i=1}^{n} |p_i - q_i| = \sum_{i=n+1}^{N} |p_i - q_i|$. We then have

$$\begin{aligned}
& |\mathrm{E}[u] - \mathrm{E}[v]| = \left| \sum_{i=1}^{N} u_i (p_i - q_i) \right| = \left| \sum_{i=1}^{n} u_i (p_i - q_i) + \sum_{i=n+1}^{N} u_i (p_i - q_i) \right| \\
\leq \; & \max \left\{ \left| \sum_{i=1}^{n} u_i (p_i - q_i) \right|, \left| \sum_{i=n+1}^{N} u_i (p_i - q_i) \right| \right\} \leq \max \left\{ \sum_{i=1}^{n} |p_i - q_i|, \sum_{i=n+1}^{N} |p_i - q_i| \right\} \\
= \; & \frac{1}{2} \sum_{i=1}^{N} |p_i - q_i| .
\end{aligned}$$

□

Proposition 1. *Fix the number of products N and let P be a CF algorithm. Then, for all M, $r \in \{0, 1/M, \ldots, (M-1)/M\}$, $Y \in S^{N \times (1-r)M}$, $Z \in S^{N \times rM}$, $\nu \in \sigma_N$, and $n \in \{1, \ldots, N\}$,*

$$d_n^{\mathrm{RMS}}(P, \nu, Y, Z) \leq \sqrt{\frac{1}{2} d_n^{\mathrm{KL}}(P, \nu, Y, Z)}.$$

Proof. Recall that $\hat{x}_{\nu_k, x^{k-1}, Y}$ and $\hat{x}_{\nu_k, x^{k-1}, (Y,Z)}$ denote the expected ratings of product ν_k with respect to PMFs $P\left(\cdot \,\middle|\, x^{k-1}, Y, \nu\right)$ and $P\left(\cdot \,\middle|\, x^{k-1}, (Y,Z), \nu\right)$, respectively. We have

$$\begin{aligned}
& d_n^{\mathrm{RMS}}(P, \nu, Y, Z) \\
= \; & \sqrt{\frac{1}{n} \sum_{k=1}^{n} \mathrm{E}\left[\left(\hat{x}_{\nu_k, x^{k-1}, Y} - \hat{x}_{\nu_k, x^{k-1}, (Y,Z)} \right)^2 \right]} \\
\leq \; & \sqrt{\frac{1}{n} \sum_{k=1}^{n} \mathrm{E}\left[\left(\frac{1}{2} \left\| P\left(\cdot \mid x^{k-1}, Y, \nu\right) - P\left(\cdot \mid x^{k-1}, (Y,Z), \nu\right) \right\|_1 \right)^2 \right]} \\
\leq \; & \sqrt{\frac{1}{2n} \sum_{k=1}^{n} \mathrm{E}\left[D\left(P\left(\cdot \mid x^{k-1}, Y, \nu\right) \,\middle\|\, P\left(\cdot \mid x^{k-1}, (Y,Z), \nu\right) \right) \right]} \\
= \; & \sqrt{\frac{1}{2} d_n^{\mathrm{KL}}(P, \nu, Y, Z)},
\end{aligned}$$

where the first inequality follows from Lemma 1 and the second inequality follows from Pinsker's inequality. We note that related results are also shown in Gossner and Tomala (2008). □

Lemma 2. *Consider a Bernoulli random variable X and discrete random variables W_1 and W_2. Let $\hat{X}_1$ and $\hat{X}_2$ be the maximum a posteriori estimates of X upon observing W_1 and W_2, respectively.*

That is,

$$\hat{X}_1 = \operatorname*{argmax}_{x\in\{0,1\}} \Pr(X = x|W_1),$$

$$\hat{X}_2 = \operatorname*{argmax}_{x\in\{0,1\}} \Pr(X = x|W_2).$$

Then,

$$\Pr\left(\hat{X}_1 = X\right) - \Pr\left(\hat{X}_2 = X\right) \leq \sqrt{\mathrm{E}[(\mathrm{E}[X|W_1] - \mathrm{E}[X|W_2])^2]}.$$

Proof.

$$\begin{aligned}
& \Pr\left(\hat{X}_1 = X\right) - \Pr\left(\hat{X}_2 = X\right) \\
= & \sum_{w_1} \Pr(W_1 = w_1) \max_{x\in\{0,1\}} \Pr(X = x|W_1 = w_1) - \sum_{w_2} \Pr(W_2 = w_2) \max_{x\in\{0,1\}} \Pr(X = x|W_2 = w_2) \\
= & \sum_{w_1,w_2} \Pr(W_1 = w_1, W_2 = w_2) \left(\max_x \Pr(X = x|W_1 = w_1) - \max_x \Pr(X = x|W_2 = w_2)\right) \\
\leq & \sum_{w_1,w_2} \Pr(W_1 = w_1, W_2 = w_2) \left|\Pr(X = 1|W_1 = w_1) - \Pr(X = 1|W_2 = w_2)\right| \\
\leq & \sqrt{\sum_{w_1,w_2} \Pr(W_1 = w_1, W_2 = w_2) \left(\Pr(X = 1|W_1 = w_1) - \Pr(X = 1|W_2 = w_2)\right)^2} \\
= & \sqrt{\mathrm{E}\left[\left(\mathrm{E}[X|W_1] - \mathrm{E}[X|W_2]\right)^2\right]}.
\end{aligned}$$

The first inequality follows from a simple arithmetic argument, and the second inequality follows from Jensen's inequality. □

Proposition 2. *Fix the number of products N. Let P be a CF algorithm and $\mathbf{S} = \{0, 1\}$. Then, for all M, $r \in \{0, 1/M, \ldots, (M-1)/M\}$, $Y \in S^{N\times(1-r)M}$, $Z \in S^{N\times rM}$, $\nu \in \sigma_N$, and $n \in \{1, \ldots, N\}$,*

$$d_n^{\mathrm{B}}(P, \nu, Y, Z) \leq d_n^{\mathrm{RMS}}(P, \nu, Y, Z).$$

Proof. Recall that $\hat{x}_{\nu_k,x^{k-1},Y}$ and $\hat{x}_{\nu_k,x^{k-1},(Y,Z)}$ denote the binary predictions on product ν_k with

respect to PMFs $P\left(\cdot \,\middle|\, x^{k-1}, Y, \nu\right)$ and $P\left(\cdot \,\middle|\, x^{k-1}, (Y,Z), \nu\right)$, respectively. We have

$$\begin{aligned}
& d_n^{\mathrm{B}}(P, \nu, Y, Z) \\
= \; & \frac{1}{n} \sum_{k=1}^{n} \left(\Pr\left(\mathbf{x}_{\nu_k} = \hat{x}_{\nu_k, x^{k-1}, Y}\right) - \Pr\left(\mathbf{x}_{\nu_k} = \hat{x}_{\nu_k, x^{k-1}, (Y,Z)}\right) \right) \\
\leq \; & \sqrt{\frac{1}{n} \sum_{k=1}^{n} \left(\Pr\left(\mathbf{x}_{\nu_k} = \hat{x}_{\nu_k, x^{k-1}, Y}\right) - \Pr\left(\mathbf{x}_{\nu_k} = \hat{x}_{\nu_k, x^{k-1}, (Y,Z)}\right) \right)^2} \\
\leq \; & \sqrt{\frac{1}{n} \sum_{k=1}^{n} \mathrm{E}\left[\left(\hat{x}_{\nu_k, x^{k-1}, Y} - \hat{x}_{\nu_k, x^{k-1}, (Y,Z)} \right)^2 \right]} \\
= \; & d_n^{\mathrm{RMS}}(P, \nu, Y, Z).
\end{aligned}$$

The first inequality follows from Jensen's inequality and the second inequality follows from Lemma 2. □

A.2 Results for Linear and Asymptotically Linear Collaborative Filtering Algorithms

Theorem 1 provides a distortion bound for linear CF algorithms. Theorem 2 provides a distortion bound for asymptotically linear CF algorithms. Lemmas 3 and 4 help prove it. Theorem 3 states a relationship between consistent and asymptotically CF algorithms. Lemmas 5, 6, and 7 help prove it.

Theorem 1. *Fix the number of products N and let P be a linear CF algorithm. Then, for all M, $r \in \{0, 1/M, \ldots, (M-1)/M\}$, $Y \in S^{N \times (1-r)M}$, $Z \in S^{N \times rM}$, $\nu \in \sigma_N$, and $n \in \{1, \ldots, N\}$,*

$$d_n^{\mathrm{KL}}(P, \nu, Y, Z) \leq \frac{1}{n} \ln \frac{1}{1-r}.$$

Proof. We have

$$\begin{aligned}
& d_n^{\mathrm{KL}}(P, \nu, Y, Z) \\
= \; & \frac{1}{n} \sum_{k=1}^{n} \mathrm{E}\left[D\left(P\left(\cdot \,\middle|\, x^{k-1}, Y, \nu \right) \,\middle\|\, P\left(\cdot \,\middle|\, x^{k-1}, (Y,Z), \nu \right) \right) \right] \\
\leq \; & \frac{1}{n} \sum_{k=1}^{N} \mathrm{E}\left[D\left(P\left(\cdot \,\middle|\, x^{k-1}, Y, \nu \right) \,\middle\|\, P\left(\cdot \,\middle|\, x^{k-1}, (Y,Z), \nu \right) \right) \right] \\
= \; & \frac{1}{n} D\left(P\left(\cdot \,|\, Y, \nu \right) \| \, P\left(\cdot \,|\, (Y,Z), \nu \right) \right).
\end{aligned}$$

The last equality follows from the chain rule of KL divergence: that is, given two joint distributions

s and t over random variables $z_1, \ldots, z_n$, it holds that

$$D(s \,\|\, t) = \sum_{k=1}^{n} \mathrm{E}\left[D(s(z_k|z_1, \ldots, z_{k-1}) \,\|\, t(z_k|z_1, \ldots, z_{k-1}))\right],$$

where expectation is taken with $z_1, \ldots, z_n$ distributed according to s.

For any $\mathbf{x} \in \mathbf{S}^N$, since P is linear, we have

$$\frac{P(\mathbf{x} \mid Y)}{P(\mathbf{x} \mid (Y, Z))} = \frac{P(\mathbf{x} \mid Y)}{(1-r)P(\mathbf{x} \mid Y) + rP(\mathbf{x} \mid Z)} \leq \frac{1}{1-r}.$$

Then,

$$D\left(P(\cdot \mid Y) \,\|\, P(\cdot \mid (Y, Z))\right) = \sum_{\mathbf{x} \in \mathbf{S}^N} P(\mathbf{x} \mid Y) \ln \frac{P(\mathbf{x} \mid Y)}{P(\mathbf{x} \mid (Y, Z))} \leq \sum_{\mathbf{x} \in \mathbf{S}^N} P(\mathbf{x} \mid Y) \ln \frac{1}{1-r} = \ln \frac{1}{1-r}.$$

Hence,

$$d_n^{\mathrm{KL}}(P, \nu, Y, Z) \leq \frac{1}{n} D\left(P(\cdot \mid Y) \,\|\, P(\cdot \mid (Y, Z))\right) \leq \frac{1}{n} \ln \frac{1}{1-r}.$$

□

Lemma 3. *Let $\{\mu_m\}$ and $\{\nu_m\}$ be two sequences of random PMFs over a fixed finite sample space Ω. If for all $\epsilon > 0$, $\Pr\left(D\left(\mu_m \,\|\, \nu_m\right) \geq \epsilon\right) \to 0$, then for all $\epsilon > 0$,*

$$\Pr\left(\sum_{\omega \in \Omega} \mu_m(\omega) \left|\log \frac{\mu_m(\omega)}{\nu_m(\omega)}\right| \geq \epsilon\right) \to 0.$$

Proof. For $\epsilon > 0$ and m, we denote by $A_{m,\epsilon}$ the event that for all $\omega \in \Omega$, at least one of the following holds: $\left|\log\left(\mu_m(\omega)/\nu_m(\omega)\right)\right| \leq \epsilon$ and $\max\{\mu_m(\omega), \nu_m(\omega)\} \leq \epsilon$. We now prove that for all $\epsilon > 0$, $\Pr\left(A_{m,\epsilon}\right) \to 1$. To see this, for any given $\epsilon > 0$, we let δ_ϵ be in $(0, \epsilon(1 - e^{-\epsilon}))$ and denote by B_{m,δ_ϵ} the event that for all $\omega \in \Omega$, $|\mu_m(\omega) - \nu_m(\omega)| \leq \delta_\epsilon$. We now show that $B_{m,\delta_\epsilon} \subset A_{m,\epsilon}$. If $|\mu_m(\omega) - \nu_m(\omega)| \leq \delta_\epsilon, \forall\, \omega$, then for any ω' such that $\max\{\mu_m(\omega'), \nu_w(\omega')\} > \epsilon$, we have $\min\{\mu_m(\omega'), \nu_m(\omega')\} > \epsilon - \delta_\epsilon$. This implies

$$\begin{aligned} &\left|\log \frac{\mu_m(\omega')}{\nu_m(\omega')}\right| = \max\left\{\log \frac{\mu_m(\omega')}{\nu_m(\omega')}, \log \frac{\nu_m(\omega')}{\mu_m(\omega')}\right\} \\ \leq\ &\max\left\{\log\left(\frac{|\mu_m(\omega') - \nu_m(\omega')|}{\nu_m(\omega')} + 1\right), \log\left(\frac{|\nu_m(\omega') - \mu_m(\omega')|}{\mu_m(\omega')} + 1\right)\right\} < \log\left(\frac{\delta_\epsilon}{\epsilon - \delta_\epsilon} + 1\right) \leq \epsilon, \end{aligned}$$

where the last inequality follows from our choice of δ_ϵ. Hence, $B_{m,\delta_\epsilon} \subset A_{m,\epsilon}$ and for any ϵ and a corresponding δ_ϵ, we have $\Pr(A_{m,\epsilon}) \geq \Pr(B_{m,\delta_\epsilon}) \to 1$, where convergence follows from Pinsker's inequality.

For each m and each realization of μ_m and ν_m, we let $\Omega_m = \{\omega \in \Omega : \mu_m(\omega) \leq \nu_m(\omega)\}$, let

$$\tau_m = \sum_{\omega \in \Omega_m} \mu_m(\omega) \log \frac{\mu_m(\omega)}{\nu_m(\omega)},$$

and for $\epsilon > 0$, denote by $C_{m,\epsilon}$ the event that $|\tau_m| \leq \epsilon$. We now show that for any $\epsilon > 0$, $A_{m,\gamma_\epsilon} \subset C_{m,\epsilon}$ where $\gamma_\epsilon \in (0, \min\{\epsilon/|\Omega|, 1/e\}]$ satisfies $\gamma_\epsilon |\log \gamma_\epsilon| \leq \epsilon/|\Omega|$. To see this, we first let $\Omega^1_{m,\gamma_\epsilon} = \{\omega \in \Omega : |\log(\mu_m(\omega)/\nu_m(\omega))| \leq \gamma_\epsilon\}$ and $\Omega^2_{m,\gamma_\epsilon} = \{\omega \in \Omega : \max\{\mu_m(\omega), \nu_m(\omega)\} \leq \gamma_\epsilon\} \backslash \Omega^1_{m,\gamma_\epsilon}$. Note that $\Omega^1_{m,\gamma_\epsilon} \cap \Omega^2_{m,\gamma_\epsilon} = \emptyset$. If for all $\omega \in \Omega$, $|\log(\mu_m(\omega)/\nu_m(\omega))| \leq \gamma_\epsilon$ or $\max\{\mu_m(\omega), \nu_m(\omega)\} \leq \gamma_\epsilon$, then $\Omega^1_{m,\gamma_\epsilon} \cup \Omega^2_{m,\gamma_\epsilon} = \Omega$. This implies

$$\begin{aligned}
|\tau_m| &= \sum_{\omega \in \Omega_m \cap \Omega^1_{m,\gamma_\epsilon}} \mu_m(\omega) \left|\log \frac{\mu_m(\omega)}{\nu_m(\omega)}\right| + \sum_{\omega \in \Omega_m \cap \Omega^2_{m,\gamma_\epsilon}} \mu_m(\omega) \left|\log \frac{\mu_m(\omega)}{\nu_m(\omega)}\right| \\
&\leq |\Omega^1_{m,\gamma_\epsilon}|\gamma_\epsilon + \sum_{\omega \in \Omega_m \cap \Omega^2_{m,\gamma_\epsilon}} \mu_m(\omega) |\log \mu_m(\omega)| \\
&\leq |\Omega^1_{m,\gamma_\epsilon}|\gamma_\epsilon + |\Omega^2_{m,\gamma_\epsilon}|\gamma_\epsilon |\log \gamma_\epsilon| \\
&\leq |\Omega^1_{m,\gamma_\epsilon}|\frac{\epsilon}{|\Omega|} + |\Omega^2_{m,\gamma_\epsilon}|\frac{\epsilon}{|\Omega|} \\
&= \epsilon.
\end{aligned}$$

The first inequality follows from the definitions of $\Omega^1_{m,\gamma_\epsilon}$ and Ω_m. The second inequality follows from the definition of $\Omega^2_{m,\gamma_\epsilon}$ and that $\gamma_\epsilon \leq 1/e$. The third inequality follows from other constraints on γ_ϵ. Hence, for any $\epsilon > 0$ and a corresponding γ_ϵ satisfying the aforesaid constraints, $A_{m,\gamma_\epsilon} \subset C_{m,\epsilon}$.

Note that for all m and all realizations of μ_m and ν_m,

$$\sum_{\omega \in \Omega} \mu_m(\omega) \left|\log \frac{\mu_m(\omega)}{\nu_m(\omega)}\right| = D(\mu_m \,\|\, \nu_m) - 2\tau_m.$$

Hence, for any $\epsilon > 0$, a corresponding γ_ϵ, and any m,

$$\begin{aligned}
&\Pr\left(\sum_{\omega \in \Omega} \mu_m(\omega) \left|\log \frac{\mu_m(\omega)}{\nu_m(\omega)}\right| \geq 3\epsilon\right) \\
&= \Pr\left(D(\mu_m \,\|\, \nu_m) - 2\tau_m \geq 3\epsilon, A^c_{m,\gamma_\epsilon}\right) + \Pr\left(D(\mu_m \,\|\, \nu_m) - 2\tau_m \geq 3\epsilon, A_{m,\gamma_\epsilon}\right) \\
&\leq \Pr\left(A^c_{m,\gamma_\epsilon}\right) + \Pr\left(D(\mu_m \,\|\, \nu_m) - 2\tau_m \geq 3\epsilon, C_{m,\epsilon}\right) \\
&\leq \Pr\left(A^c_{m,\gamma_\epsilon}\right) + \Pr\left(D(\mu_m \,\|\, \nu_m) \geq \epsilon\right) \to 0.
\end{aligned}$$

Here, A^c_{m,γ_ϵ} denotes the complement of A_{m,γ_ϵ}. The last inequality follows from the definition of $C_{m,\epsilon}$. Convergence follows from our original assumption and that $\Pr(A_{m,\gamma_\epsilon}) \to 1$. □

Lemma 4. *Let $\{\mu_m\}$, $\{\nu_m\}$, and $\{\chi_m\}$ be three sequences of random PMFs over a fixed finite sample space Ω. Suppose that for all $\epsilon > 0$, $\Pr\left(D\left(\mu_m \,\|\, \nu_m\right) \geq \epsilon\right) \to 0$. Further, suppose there*

exists $b > 0$ such that for all m, realizations of χ_m and μ_m, and $\omega \in \Omega$, $\chi_m(\omega)/\mu_m(\omega) \leq b$. Then, for all $\epsilon > 0$, $\Pr\left(\left|D\left(\chi_m \,\|\, \mu_m\right) - D\left(\chi_m \,\|\, \nu_m\right)\right| \geq \epsilon\right) \to 0$.

Proof. For any $\epsilon > 0$ and m,

$$\begin{aligned}
& \Pr\left(\left|D\left(\chi_m \,\|\, \mu_m\right) - D\left(\chi_m \,\|\, \nu_m\right)\right| \geq \epsilon\right) \\
= \; & \Pr\left(\left|\sum_{\omega\in\Omega} \chi_m(\omega) \log \frac{\mu_m(\omega)}{\nu_m(\omega)}\right| \geq \epsilon\right) \\
\leq \; & \Pr\left(\sum_{\omega\in\Omega} \frac{\chi_m(\omega)}{\mu_m(\omega)} \left|\mu_m(\omega) \log \frac{\mu_m(\omega)}{\nu_m(\omega)}\right| \geq \epsilon\right) \\
\leq \; & \Pr\left(b \sum_{\omega\in\Omega} \left|\mu_m(\omega) \log \frac{\mu_m(\omega)}{\nu_m(\omega)}\right| \geq \epsilon\right) \to 0,
\end{aligned}$$

where convergence follows from Lemma 3. □

Theorem 2. *Fix the number of products N and a set Ψ of PMFs over S^N. Let P be a CF algorithm asymptotically linear with respect to Ψ. Then, for all $\psi, \phi \in \Psi$, $r \in [0, 1)$, $\nu \in \sigma_N$, $n \in \{1, \ldots, N\}$, and $\epsilon > 0$,*

$$\lim_{m\to\infty} \Pr\left(d_n^{\text{KL}}(P, \nu, Y_m, Z_m) \geq \frac{1}{n} \ln \frac{1}{1-r} + \epsilon\right) = 0,$$

where, for each m, $Y_m = (y^1, \ldots, y^{m-l}) \in S^{N\times(m-l)}$, $Z_m = (z^1, \ldots, z^l) \in S^{N\times l}$, $l \sim$ Binomial(m, r), and $y^1, \ldots, y^{m-l} \sim \psi$ and $z^1, \ldots, z^l \sim \phi$ are i.i.d. sequences.

Proof. Let finite sample space $\Omega = \mathbf{S}^N$. For each m, let random PMFs $\mu_m = (1-r)P\left(\cdot \,|\, Y_m\right) + rP\left(\cdot \,|\, Z_m\right)$, $\nu_m = P\left(\cdot \,|\, (Y_m, Z_m)\right)$, and $\chi_m = P\left(\cdot \,|\, Y_m\right)$. By the definition of asymptotically linear CF algorithms, for all $\epsilon > 0$, $\Pr\left(D\left(\mu_m \,\|\, \nu_m\right) \geq \epsilon\right) \to 0$. Clearly, for all m, realizations of χ_m and μ_m, and $\mathbf{s} \in \mathbf{S}^N$, $\chi_m(\mathbf{s})/\mu_m(\mathbf{s}) \leq 1/(1-r)$. Hence, for all $\epsilon > 0$,

$$\begin{aligned}
& \Pr\left(d_n^{\text{KL}}(P, \nu, Y_m, Z_m) \geq \frac{1}{n} \ln \frac{1}{1-r} + \epsilon\right) \\
\leq \; & \Pr\left(D\left(P\left(\cdot\,|Y_m\right) \,\|\, P\left(\cdot\,|(Y_m, Z_m)\right)\right) \geq \ln \frac{1}{1-r} + n\epsilon\right) \\
\leq \; & \Pr\left(D\left(P\left(\cdot\,|Y_m\right) \,\|\, P\left(\cdot\,|(Y_m, Z_m)\right)\right) \geq D\left(P\left(\cdot\,|Y_m\right) \,\|\, (1-r)P\left(\cdot\,|Y_m\right) + rP\left(\cdot\,|Z_m\right)\right) + n\epsilon\right) \\
\leq \; & \Pr\left(\left|D(\chi_m \,\|\, \mu_m) - D(\chi_m \,\|\, \nu_m)\right| \geq n\epsilon\right) \to 0.
\end{aligned}$$

The second inequality holds because $D\left(P\left(\cdot\,|Y_m\right) \,\|\, (1-r)P\left(\cdot\,|Y_m\right) + rP\left(\cdot\,|Z_m\right)\right) \leq \ln(1/(1-r))$ and convergence follows from Lemma 4. □

Lemma 5. *Let $U \subset \Re^k$ be a compact set. Consider a fixed vector $u \in U$ and a sequence of random vectors $\{u_m\}$ for which $\Pr\left(u_m \in U\right) \to 1$. For any continuous function $f : U \to \Re$, if $\Pr\left(\|u_m - u\|_1 \geq \epsilon\right) \to 0$ for all $\epsilon > 0$, then $\Pr\left(|f(u_m) - f(u)| \geq \epsilon\right) \to 0$ for all $\epsilon > 0$.*

Proof. Because the continuous function f defined on compact set U is uniformly continuous, for each $\epsilon > 0$, there exists $\delta > 0$ such that for any $v, v' \in U$, $\|v - v'\|_1 \leq \delta$ implies that $|f(v) - f(v')| \leq \epsilon$. Hence, for all $\epsilon > 0$,

$$\begin{aligned} & \Pr\left(|f(u_m) - f(u)| \leq \epsilon\right) \\ \geq\ & \Pr\left(|f(u_m) - f(u)| \leq \epsilon, u_m \in U\right) \\ \geq\ & \Pr\left(\|u_m - u\|_1 \leq \delta, u_m \in U\right) \to 1. \end{aligned}$$

□

Lemma 6. *Fix a finite sample space Ω. Let $\{\mu_m\}$ and $\{\nu_m\}$ be two sequences of random PMFs over Ω and let μ be a fixed PMF over Ω. If for all $\epsilon > 0$, $\Pr\left(D\left(\mu_m \,\|\, \mu\right) \geq \epsilon\right) \to 0$ and $\Pr\left(D\left(\nu_m \,\|\, \mu\right) \geq \epsilon\right) \to 0$, then for all $\epsilon > 0$, $\Pr\left(D\left(\mu_m \,\|\, \nu_m\right) \geq \epsilon\right) \to 0$.*

Proof. We first identify the support of μ. Without loss of generality, we let $\mu(\omega_i) = 0, \forall\, 1 \leq i \leq l$ and $\mu(\omega_i) > \delta, \forall\, l < i \leq |\Omega|$ for some $l \geq 0$ and $\delta > 0$. In the following, we represent PMFs as vectors in $\Re^{|\Omega|}$. To this end, we define a set $T = \{\left(t_1, \ldots, t_{|\Omega|}\right) : \sum_i t_i = 1, \forall\, i \leq l, t_i = 0, \forall\, j > l, t_j \geq \delta\}$. Let compact set $U = T \times T$. We let $u = (\mu, \mu) \in U$ and define a sequence of random vectors $\{u_m\}$ where each $u_m = (\mu_m, \nu_m)$. Let continuous function $f : U \to \Re$ be the KL divergence $D(\cdot \,\|\, \cdot)$. By examining the absolute continuity of μ_m and ν_m with respect to μ and applying Pinsker's inequality, we have $\Pr(u_m \in U) \to 1$ and further, for all $\epsilon > 0$, $\Pr(\|u_m - u\|_1 \geq \epsilon) \to 0$. Hence, for all $\epsilon > 0$, by Lemma 5, $\Pr\left(D\left(\mu_m \,\|\, \nu_m\right) \geq \epsilon\right) = \Pr\left(|D\left(\mu_m \,\|\, \nu_m\right) - D\left(\mu \,\|\, \mu\right)| \geq \epsilon\right) = \Pr\left(|f(u_m) - f(u)| \geq \epsilon\right) \to 0$. □

Lemma 7. *Fix a finite sample space Ω. Let $\{\mu_m\}$ and $\{\nu_m\}$ be two sequences of random PMFs over Ω and let μ and ν be two fixed PMFs over Ω. If for all $\epsilon > 0$, $\Pr\left(D\left(\mu_m \,\|\, \mu\right) \geq \epsilon\right) \to 0$ and $\Pr\left(D\left(\nu_m \,\|\, \nu\right) \geq \epsilon\right) \to 0$, then for all $r \in [0, 1]$ and $\epsilon > 0$,*

$$\Pr\left(D\left(((1-r)\mu_m + r\nu_m) \,\|\, ((1-r)\mu + r\nu)\right) \geq \epsilon\right) \to 0.$$

Proof. The proof here is similar to that for Lemma 6. For a fixed r, let PMF $\chi = (1-r)\mu + r\nu$. Without loss of generality, we let $\chi(\omega_i) = 0, \forall\, 1 \leq i \leq l$ and $\chi(\omega_i) > \delta, \forall\, l < i \leq |\Omega|$ for some $l \geq 0$ and $\delta > 0$. In the following, we represent PMFs as vectors in $\Re^{|\Omega|}$. To this end, we define a set $T = \{\left(t_1, \ldots, t_{|\Omega|}\right) : \sum_i t_i = 1, \forall\, i \leq l, t_i = 0, \forall\, j > l, t_j \geq \delta\}$. Let compact set $U = T \times T$. We let $u = ((1-r)\mu + r\nu, (1-r)\mu + r\nu) \in U$ and define a sequence of random vectors $\{u_m\}$ where each $u_m = ((1-r)\mu_m + r\nu_m, (1-r)\mu + r\nu)$. Let continuous function $f : U \to \Re$ be the KL divergence $D(\cdot \,\|\, \cdot)$. By examining the absolute continuity of μ_m with respect to μ and that of ν_m with respect to ν and applying Pinsker's inequality, we have $\Pr(u_m \in U) \to 1$, and for all $\epsilon > 0$,

$\Pr(\|u_m - u\|_1 \geq \epsilon) \to 0$. Hence, for all $\epsilon > 0$, by Lemma 5,

$$\begin{aligned} & \Pr\left(D\left(((1-r)\mu_m + r\nu_m) \,\|\, ((1-r)\mu + r\nu)\right) \geq \epsilon\right) \\ = \; & \Pr\left(\left|D\left(((1-r)\mu_m + r\nu_m) \,\|\, ((1-r)\mu + r\nu)\right) - D\left(((1-r)\mu + r\nu) \,\|\, ((1-r)\mu + r\nu)\right)\right| \geq \epsilon\right) \\ = \; & \Pr\left(|f(u_m) - f(u)| \geq \epsilon\right) \to 0. \end{aligned}$$

□

Theorem 3. *Any probabilistic CF algorithm consistent with respect to an identifiable and convex set Φ is asymptotically linear with respect to Φ.*

Proof. We use the notation in the definition of asymptotically linear CF algorithms in Section 4.3 and Lemmas 6 and 7. Fix an algorithm P consistent with respect to Φ. Let finite sample space $\Omega = \mathbf{S}^N$. For each m, let random PMFs $\mu_m = P(\cdot\,|U_m)$, $\nu_m = P(\cdot\,|V_m)$, and $\chi_m = P(\cdot\,|(U_m, V_m))$. Let $\overline{P}_1$ and $\overline{P}_2$ be the type PMFs corresponding to ψ and ϕ, respectively. Let $\mu = \overline{P}_1$, $\nu = \overline{P}_2$. Fix $r \in [0,1]$. By the consistency and convexity of $\mathcal{P}$, we have for all $\epsilon > 0$, $\Pr(D(\mu_m \,\|\, \mu) \geq \epsilon) \to 0$, $\Pr(D(\nu_m \,\|\, \nu) \geq \epsilon) \to 0$, and $\Pr(D(\chi_m \,\|\, ((1-r)\mu + r\nu)) \geq \epsilon) \to 0$. By Lemma 7, for all $\epsilon > 0$, $\Pr(D(((1-r)\mu_m + r\nu_m) \,\|\, ((1-r)\mu + r\nu)) \geq \epsilon) \to 0$. Then for all $\epsilon > 0$,

$$\begin{aligned} & \Pr\left(D\left(((1-r)P(\cdot\,|U_m) + rP(\cdot\,|V_m)) \,\|\, P(\cdot\,|(U_m, V_m))\right) \geq \epsilon\right) \\ = \; & \Pr\left(D\left(((1-r)\mu_m + r\nu_m) \,\|\, \chi_m\right) \geq \epsilon\right) \to 0, \end{aligned}$$

where convergence follows from Lemma 6. □

A.3 Results for Kernel Density Estimation and Naive Bayes Algorithms

Propositions in this section pertain to KDE and NB algorithms.

Proposition 3. *Any KDE algorithm is a linear CF algorithm. Any linear CF algorithm is a KDE algorithm.*

Proof. Consider a KDE algorithm P. Given $W_1 \in S^{N \times M_1}$ and $W_2 \in S^{N \times M_2}$, for each $\mathbf{x} \in \mathbf{S}^N$,

$$\begin{aligned} & P(\mathbf{x}\,|(W_1, W_2)) \\ = \; & \frac{1}{M_1 + M_2}\left(\sum_{w \in W_1} \mathcal{K}_w(\mathbf{x}) + \sum_{w \in W_2} \mathcal{K}_w(\mathbf{x})\right) \\ = \; & \frac{M_1}{M_1 + M_2}\left(\frac{1}{M_1}\sum_{w \in W_1} \mathcal{K}_w(\mathbf{x})\right) + \frac{M_2}{M_1 + M_2}\left(\frac{1}{M_2}\sum_{w \in W_2} \mathcal{K}_w(\mathbf{x})\right) \\ = \; & \frac{M_1}{M_1 + M_2} P(\mathbf{x}\,|W_1) + \frac{M_2}{M_1 + M_2} P(\mathbf{x}\,|W_2). \end{aligned}$$

Hence, P is linear.

To show the converse, consider a linear CF algorithm P', which generates $P'(\cdot\,|W)$ based on W. A KDE algorithm where kernel $\mathcal{K}_w$ is set equal to $P(\cdot\,|\{w\})$ for each $w \in W$ is equivalent to P'. □

Proposition 4. *Fix $q \in [0,1)$. Let Φ_q be the set of all $(\overline{P}, \overline{Q}_q)$ pairs where user type distribution $\overline{P}$ over $\mathbf{S}^N$ takes the form*

$$\overline{P}(\mathbf{w}) = \sum_{l=1}^{L} \eta_l \prod_{n=1}^{N} \theta_{l,n}^{(\mathbf{w}_n)}, \forall\, \mathbf{w} \in \mathbf{S}^N, \tag{5}$$

for $L \in \mathcal{Z}_+$, $\eta \in \Delta_L$, and $\theta_{l,n} \in \Delta_{|\mathbf{S}|}$ for all $1 \le l \le L, 1 \le n \le N$, where $\Delta_k = \{(d_1, \ldots, d_k) : \forall\, i, d_i \ge 0, \sum_j d_j = 1\}$ denotes a simplex, and conditional distribution of ratings vectors $\overline{Q}_q$ conditioned on types is fixed as

$$\overline{Q}_q(w \mid \mathbf{w}) = q^{\|w\|_\circ}(1-q)^{N-\|w\|_\circ}, \text{ if } \mathbf{w} \to w, \text{ and}$$

$$\overline{Q}_q(w \mid \mathbf{w}) = 0, \text{ otherwise.}$$

Then, Φ_q is identifiable and convex. Further, the naive Bayes algorithm is consistent with respect to Φ_q.

Proof. To show that Φ_q is identifiable, we note that $\overline{Q}_q$ is fixed and hence only need to establish that distinct $\overline{P}_1$ and $\overline{P}_2$ lead to distinct ratings vector PMFs $\overline{P}_1\overline{Q}_q$ and $\overline{P}_2\overline{Q}_q$. To see this, consider $\mathbf{w} \in \mathbf{S}^N$ such that $\overline{P}_1(\mathbf{w}) \ne \overline{P}_2(\mathbf{w})$. We have

$$\left(\overline{P}_1\overline{Q}_q\right)(\mathbf{w}) = \overline{P}_1(\mathbf{w})\overline{Q}_q(\mathbf{w}|\mathbf{w}) \ne \overline{P}_2(\mathbf{w})\overline{Q}_q(\mathbf{w}|\mathbf{w}) = \left(\overline{P}_2\overline{Q}_q\right)(\mathbf{w}).$$

To show that Φ_q is convex, consider arbitrary type PMFs $\overline{P}_1, \overline{P}_2$ whose corresponding pairs are in Φ_q. For each $\lambda \in [0,1]$, their convex combination $\overline{P}_\lambda$ satisfies

$$\begin{aligned}
& \overline{P}_\lambda(\mathbf{w}) = \lambda\overline{P}_1(\mathbf{w}) + (1-\lambda)\overline{P}_2(\mathbf{w}) \\
= \;& \sum_{l=1}^{L_1} \lambda\eta_{1,l} \prod_{n=1}^{N} \theta_{1,l,n}^{(\mathbf{w}_n)} + \sum_{l=1}^{L_2} (1-\lambda)\eta_{2,l} \prod_{n=1}^{N} \theta_{2,l,n}^{(\mathbf{w}_n)} \\
= \;& \left(\sum_{l=1}^{L_1+L_2} \hat{\eta}_l \prod_{n=1}^{N} \hat{\theta}_{l,n}^{(\mathbf{w}_n)} \right)
\end{aligned}$$

for each $\mathbf{w} \in \mathbf{S}^N$, where $\hat{\eta}_l = \lambda\eta_{1,l}$ for $l \le L$, $\hat{\eta}_l = (1-\lambda)\eta_{2,l-L}$ for $l > L$, and for each n, $\hat{\theta}_{l,n} = \theta_{1,l,n}$ for $l \le L$ and $\hat{\theta}_{l,n} = \theta_{2,l-L,n}$ for $l > L$. Hence, $\overline{P}_\lambda$ takes the form in (5). As such, $(\overline{P}_\lambda, \overline{Q}_q) \in \Phi_q$ and Φ_q is convex.

We now show that the naive Bayes algorithm P is consistent with respect to Φ_q by using results in Wald (1949). We will use notation in the definition of consistent CF algorithms in Section 4.3.

We also denote by P^* the true type PMF over $\mathbf{S}^N$ and let $\mathcal{P} = \{\overline{P} : (\overline{P}, \overline{Q}) \in \Phi_q\}$ be the set of all type PMFs with corresponding pairs in Φ_q. According to Theorem 2 in Wald (1949), if

1. $\mathbf{S}^N$, $\mathcal{P}$, and P^* satisfy certain technical conditions specified in Wald (1949), and
2. There exists a constant $b > 0$ such that for all m and all realizations of $\{W_m\}$,

$$\frac{\prod_{w \in W_m} P\left(w \mid W_m\right)}{\prod_{w \in W_m} P^*(w)} \geq b,$$

then

$$\left\| P\left(\cdot \mid W_m\right) - P^* \right\|_1 \to 0, \; a.s. \tag{6}$$

We verify that condition 1 holds in our problem instance and desire to find a b that satisfies condition 2. To do so, we denote by (L^*, η^*, θ^*) the parameters corresponding to P^* and by $(\hat{L}^{W_m}, \hat{\eta}^{W_m}, \hat{\theta}^{W_m})$ the parameters corresponding to $P\left(\cdot \mid W_m\right)$. Recall that we tune the parameter τ by cross validation over a range Γ. Let τ^* be the value that we settle at. We let

$$b = \left(\min_{\tau \in \Gamma} f_L^{\tau}(L^*) \right) \left(f_\eta(\eta^*) \prod_{l,n} f_\theta(\theta^*_{l,n}) \right).$$

Because $P\left(\cdot \mid W_m\right)$ maximizes the posterior PDF, for all m and all realizations of $\{W_m\}$, we have

$$\frac{\prod_{w \in W_m} P\left(w \mid W_m\right)}{\prod_{w \in W_m} P^*(w)} \geq \frac{f_L^{\tau^*}(L^*) f_\eta(\eta^*) \prod_{l,n} f_\theta(\theta^*_{l,n})}{f_L^{\tau^*}(\hat{L}^{W_m}) f_\eta(\hat{\eta}^{W_m}) \prod_{l,n} f_\theta(\hat{\theta}^{W_m}_{l,n})} \geq b.$$

Hence, condition 2 holds, establishing (6). By the continuity of KL divergence and properties of almost sure convergence, for all $\epsilon > 0$, we have

$$\Pr\left(D\left(P\left(\cdot \mid W_m\right) \,\|\, P^*\right) \geq \epsilon \right) \to 0,$$

which implies that P is consistent with respect to Φ_q. □

B Table of Notation

Notation	Description
$\mathbf{S}$	Set of possible rating values.
S	Union of $\mathbf{S}$ and the singleton set containing the circle $\circ$.
$\mathbf{w}$ or $\mathbf{x}$	A user type.
w or x	A ratings vector.
x^k	Ratings vector in S^N that contains k ratings provided by an active user.
$p_{n,x,W}$	PMF of the rating for product n, for a user with history x, based on training data W.
$P(\cdot \mid W, \nu)$	Joint distribution of ratings generated by a CF algorithm based on training data W and product ordering ν.
σ_N	Set of permutations of $\{1, \ldots, N\}$.
d^{KL}	KL distortion.
d^{RMS}	RMS distortion.
d^{B}	Binary prediction distortion.
$\hat{x}_{n,x,W}$	Scalar prediction of rating of product n for a user with history x, based on training data W.
$\hat{d}^{\mathrm{RMS}}$	Empirical RMS distortion.
$\hat{\mathcal{E}}^{\mathrm{RMS}}$	Empirical RMS prediction error.
$\mathbf{w} \to w$	For each n, either $w_n = \mathbf{w}_n$ or $w_n = \circ$.
$\|w\|_\circ$	Number of circles in ratings vector w: $\lvert\{n : w_n = \circ\}\rvert$.