Collaborative recommendation: A robustness analysis

Michael O'Mahony, Neil Hurley, Nicholas Kushmerick, Guenole Silvestre

Computer Science Department, University College Dublin, Ireland

Abstract. Collaborative recommendation has emerged as an effective technique for personalised information access. However, there has been relatively little theoretical analysis of the conditions under which the technique is effective. To explore this issue, we analyse the *robustness* of collaborative recommendation: the ability to make recommendations despite (possibly intentional) noisy product ratings. There are two aspects to robustness: recommendation *accuracy* and *stability*. We formalise these aspects in machine learning terms and develop theoretically justified models of accuracy and stability. We then evaluate our analysis using several real-world data-sets. Our investigation is both practically relevant for enterprises wondering whether collaborative recommendation leaves their marketing operations open to attack, and theoretically interesting for the light it sheds on a comprehensive theory of collaborative recommendation.[1]

1 Introduction

Collaborative recommendation has emerged as an effective personalisation technique for a diverse array of electronic commerce and information access scenarios, e.g. [18, 6]. Such systems keep track of their customers' preferences, and use these data to offer new suggestions. Many variations have been explored, but the basic idea is as follows: to recommend items to a target customer, the system retrieves customers who have expressed similar preferences to the target, and then recommends items that were liked by the retrieved customers but not yet rated by the target.

The accuracy of various collaborative recommendation algorithms has been empirically validated for many domains, e.g. [4], and the technology has been successfully deployed in many commercial settings. However, despite some efforts [15, 5, 3], there is no general theoretical explanation of the conditions under which a particular collaborative recommendation application will succeed or fail.

[1] Submitted to ACM Transactions on Internet Technology, Special Issue of Machine Learning for the Internet; draft of May 14, 2002.

Our goal is to complement existing theoretical work by investigating the *robustness* of collaborative recommendation. In essence, robustness measures the ability of the algorithm to make good predictions in the presence of noisy data. Collaborative recommendation applications update their datasets when users enter new ratings, but there is no guarantee that these ratings reflect the user's true preferences. Indeed, since rating entry can be an onerous process, users may be careless and inaccuracies in the data must be expected.

More sinisterly, it is possible to imagine scenarios in which malicious users may be motivated to deliberately attack the recommendation system to cause it to malfunction. For example, publishers may wish to promote their work by encouraging a book recommendation system to output artificially high ratings for their publications. In some cases, the cost of perturbing the recommendation system's output may be prohibitive (e.g. only users who have purchased a book may be permitted to enter ratings). However, many collaborative recommenders are open Web services that malicious agents could easily attack. How much damage can they inflict?

There are two aspects to robustness. First, we may be concerned with recommendation *accuracy*: are the products recommended after the attack actually liked? The second issue is *stability*: does the system recommend different products after the attack (regardless of whether customers like them)?

While stability and accuracy are distinct, they are not independent. For example, if a recommender has perfect accuracy for a given task both with and without noise, then it must be perfectly stable. On other the hand, consider a product that no-one likes. The recommendation policies *recommend to no-one* and *recommend to everyone* are both perfectly stable, yet the first is always correct and the second is always wrong.

The remainder of this article is organised as follows. We begin with a discussion of collaborative recommendation and a formalisation of the notions of robustness and the perturbations with which we are concerned (Section 2). We then analyse robustness from both the accuracy and stability perspectives. Regarding accuracy, in Section 3 we formalise robustness in machine learnings terms, and introduce a novel form of class noise that models an interesting suite of attacks. We develop two models that predict the change in accuracy as a function of the number of fake ratings that have been inserted into the customer/product matrix. Regarding stability, we present a framework that describes the stability of a recommendation system subjected to various forms of attack (Section 4). In both cases, we empirically evaluate our predications against several real-world

data-sets. We conclude with a description of related work (Section 5) and with a summary of our results and a discussion of several open issues (Section 6).

2 Robustness and collaborative recommendation

Fig. 1 illustrates the essential ideas of collaborative recommendation, as well as our concerns with robustness. Collaborative filtering involves recording customer ratings for various products. For example, customer A likes product 1 but dislikes product 2.

Suppose the system wants to determine whether a particular target customer H will like product 7. To do so, the system would compare the ratings of customer H to all the other customers. For now, consider only the top portion of the matrix, customers A–H. The details depend on the particular similarity function employed, but presumably the system would determine that customers A and F are similar to H, since they tend to agree on the products they have rated in common. Recommender systems predict a rating for H based on the ratings given by the retrieved similar customers. In this case, we suppose that the system predicts that the target customer H will indeed like product 7. Collaborative filtering research over the past decade has explored a wide variety of algorithms for accurately making such predictions.

Now suppose that a competitor to product 7 decides to influence the recommender system in order that product 7 is rarely recommended. To do so, a rogue competitor could pose as a set of fake customers I–M. What ratings should the competitor promulgate from these customers in order to decrease the rating for 7? (We call such a manoeuvre a *nuke* of product 7.) Clearly, none of the fake users should like product 7; see the last column of rows I–M. In this article we describe several ways in which ratings for products 1–6 might be generated. For now, the point is simply that the rogue profiles may well influence the recommender system's prediction. For example, rogue user L is now highly similar to the target customer, which may well lead the recommender system to switch its prediction from like to dislike.

This lack of robustness can be quantified in two ways. On the one hand, we can measure the accuracy of the predicted ratings: does customer H in fact like product 7? On the other hand, we can measure the stability of the system. Even if we can not determine whether customer H likes product 7, the mere fact that the recommender system switched

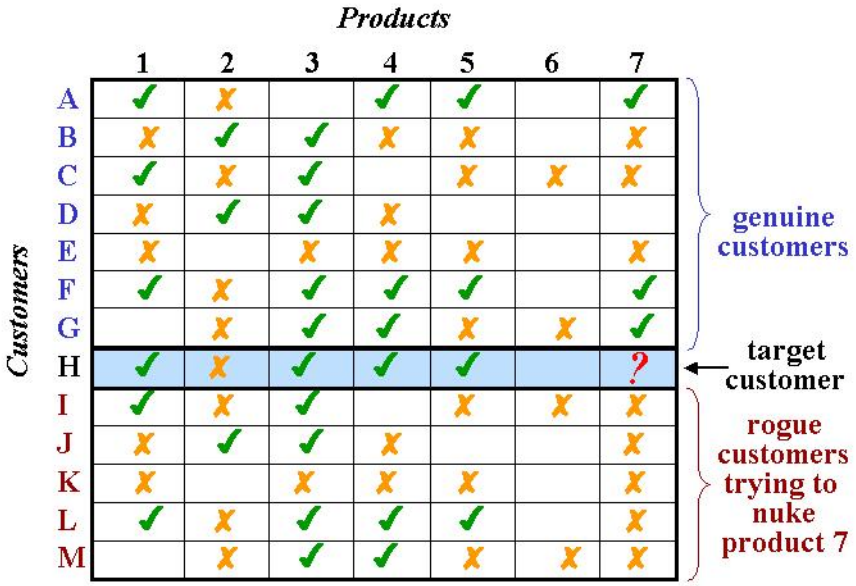

Fig. 1. A simple example of collaborative filtering, and an malicious attack designed to nuke one particular product.

its prediction may be problematic - even if the unbiased prediction was wrong!

In the limit, a malicious agent can clearly force a recommender system to behave in an arbitrary fashion, by adding to the ratings matrix enough rogue users to completely dilute the preferences of the real users. But such an attack would be extremely expensive (even assuming an open system where network bandwidth is the dominant cost), eliminating any potential gain in market share or whatever other factors lead the malicious agent to pose its attack. In this paper we are therefore interested in how the effectiveness of an attack varies with its size.

The performance of any system can only be measured with respect to some utility function which models the usefulness of a recommendation. Clearly, the utility of a recommendation is dependent on the perspective of the players involved. In collaborative recommendation systems, three types of player can be identified. First, there is the end-user, to whom the recommendations are delivered. From this perspective, the utility is dependent on how well the recommendation system follows the end-user's

personal taste. Second, there is the database owner, who is primarily interested in the throughput of the system as measured by the total number of transactions. From this perspective, the recommendation system need not actually be accurate, so long as it attracts customers to the service. Third, there are external interested parties who have an indirect interest in the transactions. For a book recommendation system, such a party might be the author or publisher of some of the books that are being recommended. Obviously, these third parties have a vested interest in the recommendations that are made for their produce, but do not have direct access to the recommendation system.

In our model of robustness, we take the end-user's perspective, and seek systems which are accurate and consistent in the recommendations that they make to end-users. Primarily, we examine how the system can be compromised by external third-parties. To such attackers, the system is a black-box which can only be viewed through the recommendations that it outputs and can only be modified by the insertion of new data through the normal user interface.

In our analysis we assume the standard k-Nearest Neighbours (k-NN) algorithm. This algorithm operates by selecting the k most similar users to the target customer, and formulates a prediction by combining the preferences of these users. Other, e.g. model-based, approaches have been tried but k-NN is reasonably accurate, widely used and easily analysed.

3 Accuracy Analysis

Our analysis of accuracy involves developing two theoretically justified models of accuracy, as a function of the size of the attack. In machine learning terms, an attack corresponds to the addition of noise to the training data. In general, this noise could be associated with either the attributes, the class or both. We focus exclusively on class noise, and defer attribute noise to future work.

The assumption of noise-free attributes corresponds to a rather strong assumption about the attacker. We do not assume that the attacker is omniscient (i.e., knows the entire ratings matrix prior to the attack). However, the attack profiles have noise-free attributes only if the attacker knows the probability distribution over the space of ratings for all products. We are modelling an attacker that therefore lies in the middle of the spectrum between ignorance and omniscience.

We are not concerned with malicious class noise as defined by [13]. Rather, we model attacks with a relatively benign noise model that we

call *biased class noise*. This model is characterised by two parameters: the noise rate β, and the class bias μ. Noise is added according to the following process. First, an instance is generated according to the underlying distribution. With probability $1-\beta$, the instance is noise-free and labelled by the target concept. Otherwise, with probability $\beta\mu$ the instance is labelled 1 and with probability $\beta(1-\mu)$ the instance is labelled 0. Note that biased class noise is not a special case of the much-studied class noise model, in which the correct class label is inverted for some fixed fraction of the training data.

The biased class noise model is natural for representing a variety of stereotypical attacks. For example, a book's author could try to force recommendations of his book by pretending to be numerous customers who all happen to like the book. We call this a PUSH attack and it corresponds to $\mu = 1$. Alternatively, the author's arch-enemy could insert fake customer profiles that all dislike the book; this NUKE attack is modeled with $\mu = 0$.

Before proceeding, we need some additional notation. We assume a d-dimensional instance space X^d, where (wlog) take X to be $[0,1]$. In the context of recommendation, each dimension corresponds to one of d products, and the value on the dimension is a numeric rating. We assume a function $\mathsf{dist}(\cdot,\cdot)$ defined over $X^d \times X^d$, but our analysis does not depend on any particular distance metric. Finally, let H be a hypothesis, C be a concept, and D be a probability distribution over X^d. The error rate of H with respect to C and D is defined as $\mathsf{err}(H,C,D) = \Pr_{x\in D}(H(x) \neq C(x))$.

3.1 Absolute accuracy

The first model extends Albert and Aha's noise-free PAC [19, 8] results for k-NN [2] to handle biased class noise. We first review these noise-free results, and then state our model as Theorem 1.

The key idea behind Albert and Aha's (hereafter: AA) analysis is that of a *sufficiently dense* sample from the instance space. Informally, a subset $S \subset X^d$ is dense if most of the points in the entire space X^d are near many points in the sample S. The terms *most*, *near* and *many* are formalised as follows: Let D be a distribution over X^d. A subset $S \subseteq X^d$ is (k,α,γ)-*dense* if, except for a subset with probability less than γ under D, for every $x \in X^d$, there exists at least k distinct points $x_1,\ldots,x_k \in S$ such that $\mathsf{dist}(x,x_i) \leq \alpha$ for each i.

Given this definition, AA derive [2, Lemma 2.2] a lower bound $\Upsilon_{\text{den}}(k, \alpha, \gamma, |S|)$ on the probability that a sample S of X^d is (k, α, γ)-dense:

$$\Upsilon_{\text{den}}(k, \alpha, \gamma, |S|) = 1 - m^d \sum_{0 \leq k' < k} \Phi_2(\rho, k', |S|),$$

where

$$\begin{aligned} m &= \left\lceil \sqrt{d}/\alpha \right\rceil, \\ \rho &= \frac{\gamma}{m^d}, \\ \Phi_2(\rho, t, s) &= \binom{t}{s} B\left(\max\left\{s/t, \rho\right\}, s, t\right), \text{ and} \\ B(p, s, t) &= p^s (1-p)^{t-s}. \end{aligned}$$

(See Appendix A for a proof.) In the spirit of the PAC model, $\Upsilon_{\text{den}}(k, \alpha, \gamma, |S|)$ is a worst-case lower bound that does not depend on the distribution D over X^d. Note also that $\Upsilon_{\text{den}}(k, \alpha, \gamma, |S|)$ varies with d, but we do not explicitly indicate this dependency because d is fixed by the learning task.

The final step in AA's analysis is to formalise the intuition that k-NN is accurate to the extent that its training sample is sufficiently dense. To describe their result, we first must constrain both the complexity of the concept being learned and the distribution over the instance space. For any constant B, let $\mathcal{D}_B$ be the class of distributions over X^d such that no point has probability exceeding B. For any constant L, let $\mathcal{C}_L$ is the set of concepts C over X^d such that C is the union of a set of regions bounded by closed hyper-curves of total length less than L.

These constraints mean that AA's results hold only for a specific class of learning tasks, and are not truly distribution-free. However, we will see that in fact the model's predictions for real-world tasks are not very sensitive to the values of L and B.

Let $C \in \mathcal{C}_L$ be a concept to be learned, and $D \in \mathcal{D}_B$ be a distribution over X^d. AA's central result [2, Theorem 3.2] is that the probability that the error of k-NN exceeds ϵ when trained on a sample S is at most $\Upsilon_{\text{den}}\left(k, \epsilon/4LB, \epsilon/2, |S|\right)$.[2]

[2] We have departed from AA in several ways. First, AA use the notation k-$\langle \alpha, \gamma \rangle$-*net*; we refer to *denseness* because our biased class noise analysis involves an analogous notion of sparseness. Second, our proof is somewhat different and therefore our bound on the denseness probability differs slightly from AA's. Most importantly, as is standard in PAC analysis, AA introduce an additional confidence parameter δ and solve $\Upsilon_{\text{den}}(k, \epsilon/4LB, \epsilon/2, |S|) > 1 - \delta$ for $|S|$, in order to show that k-NN can PAC-learn efficiently. Since robustness is orthogonal to efficiency, we ignore this part of their analysis.

The intuition behind AA's results is that k-NN is accurate when training on a sufficient number of instances. We can extend this intuition to handle biased class noise by requiring that, in addition to the sample containing enough *good* (noise-free) instances, it must also not contain too many *bad* (noisy) instances.

We begin by defining sparse subsets analogously to the definition of dense subsets. Informally, a subset $S \subset X^d$ is sparse if most of the points in the entire space X^d are near few points in the sample S. More precisely: Let D be a distribution over X^d. A subset $S \subseteq X^d$ is (k, α, γ)-*sparse* if, except for a subset with probability less than γ under D, for every x there exists at most k points x_i such that $\mathsf{dist}(x, x_i) \leq \alpha$.

Appendix A proves the following lower bound $\Upsilon_{\text{spa}}(k, \alpha, \gamma, |S|)$ on the probability that a sample S of X^d is (k, α, γ)-sparse:

$$\Upsilon_{\text{spa}}(k, \alpha, \gamma, |S|) \;=\; 1 - m^d \sum_{\langle k', k'' \rangle \in \mathcal{K}} \Phi_3(\rho, k', k'', |S|),$$

where m and ρ are as defined above,

$$\begin{aligned}
\Phi_3(\rho, r, s, t) &= \binom{t}{r} \binom{t-r}{s} T\left(\max\left\{r/t, \rho\right\}, r, s/t, s, t\right), \\
T(p, r, q, s, t) &= p^r q^s (1-p-q)^{t-r-s}, \text{ and} \\
\mathcal{K} &= \left\{\langle k', k'' \rangle | 0 \leq k', k'' \leq |S| \wedge k < k' + k'' \leq |S|\right\}.
\end{aligned}$$

To complete our analysis, we observe that the accuracy of k-NN with training data S is equal to the probability that S contains enough *good* instances and not too many *bad* instances. For k-NN, *enough* and *not too many* mean that most of the instances should have at least $\lceil k/2 \rceil$ good neighbours and at most $\lfloor k/2 \rfloor$ bad neighbours.

Let $S_{\text{real}} \subset S$ be the examples corresponding to genuine users, and $S_{\text{attack}} = S \setminus S_{\text{real}}$ be the examples comprising the attack. Of course, we can not know S_{real} and S_{attack} exactly, but we do know that $|S_{\text{real}}| = (1-\beta)|S|$ and $|S_{\text{attack}}| = \beta|S|$ (where β is the size of the attack), which is sufficient for our analysis.

Furthermore, some of the noisy instances may in fact be correctly labelled. Let f be the fraction of the instance space labelled 1, and let μ be the class noise bias. Then a fraction $f\mu + (1-f)(1-\mu)$ of the noisy instances are in fact correctly labelled. Let $S_{\text{good}} \supseteq S_{\text{real}}$ be the instances that are actually labelled correctly, and $S_{\text{bad}} = S \setminus S_{\text{good}} \subseteq S_{\text{attack}}$ be the instances that are actually labelled incorrectly. Again, we can not know S_{good} or S_{bad} but we do know that the number of noise-free instances is

$$|S_{\text{good}}| = (1-\beta)|S| + \beta|S|(f\mu + (1-f)(1-\mu)) \geq |S_{\text{real}}|,$$

and the number of noisy instances is

$$|S_{\text{bad}}| = \beta|S|(1 - f\mu - (1-f)(1-\mu)) \leq |S_{\text{attack}}|.$$

If $\lambda = \beta(\mu + f - 2\mu f)$ is the effective attack size, then $|S_{\text{good}}| = (1-\lambda)|S|$ and $|S_{\text{bad}}| = \lambda|S|$.

We require that both S_{good} be $(\lceil k/2 \rceil, \alpha, \gamma)$-dense and S_{bad} be $(\lfloor k/2 \rfloor, \alpha, \gamma)$-sparse. Since these events are independent, the probability of their conjunction is the product of their probabilities. Therefore, if we can determine appropriate values for the distance thresholds α_1 and α_2 and probability thresholds γ_1 and γ_2, then we have that the accuracy of k-NN when training on S with biased class noise is at least

$$\Upsilon_{\text{den}}(\lceil k/2 \rceil, \alpha_1, \gamma_1, |S_{\text{good}}|) \cdot \Upsilon_{\text{spa}}(\lfloor k/2 \rfloor, \alpha_2, \gamma_2, |S_{\text{bad}}|).$$

In Appendix A we prove the following theorem.

Theorem 1 (Absolute accuracy). *The following holds for any ϵ, β, μ, d, k, L, B, $C \in \mathcal{C}_L$ and $D \in \mathcal{D}_B$. Let S be a sample of X^d according to D with biased class noise rate β. Then we have that*

$$\Pr\left[\mathsf{err}(k\text{-NN}(S), C, D) < \epsilon\right] \geq \Upsilon_{\text{den}}\left(\left\lceil \frac{k}{2} \right\rceil, \frac{\epsilon}{4LB}, \frac{\epsilon}{4}, (1-\lambda)|S|\right) \cdot \Upsilon_{\text{spa}}\left(\left\lfloor \frac{k}{2} \right\rfloor, \frac{\epsilon}{4LB}, \frac{\epsilon}{4}, \lambda|S|\right),$$

where $\lambda = \beta(\mu + f - 2\mu f)$, and f is the fraction of X^d labelled 1 by C.

To summarise, Theorem 1 yields a worst-case lower bound on the accuracy of k-NN under biased class noise. On the positive side, this bound is *absolute* in the sense that it predicts (a probabilistic bound on) the actual error $\mathsf{err}(k\text{-NN}(S), C, D)$ as a function of the sample size $|S|$, noise rate β, and other parameters. In other words, the term *absolute* draws attention to the fact that this model takes account of the actual position along the learning curve. Unfortunately, like most PAC analyses, its bound is very weak (though still useful in practise; see Section 3.3).

3.2 Approximate relative accuracy

In contrast, the second model does not rely on a worst-case analysis and so makes tighter predictions than the first model. On the other hand, the model is only *approximate* because it makes two assumptions. First, it assumes that the training sample is large enough that the learning curve

has *flattened out*. Second, it assumes that, at this flat part of the learning curve, k-NN achieves perfect accuracy except possibly on the boundary of the target concept. We call this second model *approximate* to draw attention to these assumptions, and *relative* to note specifically that it does not predict error on an absolute scale.

To formalise these assumptions, let S be a training sample drawn from the distribution D over X^d, and let C be the target concept. Let S' be the fraction $1-\beta$ of the instances in S that were (correctly) labelled by C during the biased class noise process. Let D' be the distribution that is proportional to D except that $D'[x] = 0$ for all points x on the boundary between C and $X^d \setminus C$. The assumptions of the second model can be expressed as:

$$\mathsf{err}(k\text{-NN}(S'), C, D') = 0 \tag{1}$$

Given this assumption, we can predict the error of k-NN as follows. To classify an instance x using a training set S, k-NN predicts the majority class of the k instances $x_1, \ldots, x_k \in S$ that are closest to x. To classify x correctly, at least $\lceil k/2 \rceil$ of these k instances must have the correct class.

If we randomly draw from D a point $x \in X^d$, there are two cases: either $C(x) = 1$ (which happens with probability f), or $C(x) = 0$ (which happens with probability $1-f$), where as above f is the probability under D that $C(x) = 1$.

In the first case, we need to have at least $\lceil k/2 \rceil$ successes out of k trials in a Bernoulli process where the probability of success is equal to the probability that a neighbour x_i of x will be labelled 1. We can calculate this probability as $(1-\beta)+\beta\mu$. The first term is the probability that x_i is labelled 1 and $x_i \in S'$; by (1), we know that this probability is $1-\beta$. The second term is the probability that x_i is labelled 1 and $x_i \notin S'$. By the definition of the biased class noise process, we know that this probability is $\beta\mu$.

In the second case, again we need at least $\lceil k/2 \rceil$ successes, but with success probability $(1-\beta)+\beta(1-\mu)$, the probability that a neighbour x_i of x will be labelled 0. The first term is the probability that x_i is labelled 0 and $x_i \in S'$, and by (1) this happens with probability $1-\beta$. The second term is the probability that x_i is labelled 0 and $x_i \notin S'$, which occurs with probability $\beta(1-\mu)$.

The following theorem follows from this discussion.

Theorem 2 (Approximate relative accuracy). *The following holds for any β, μ, d, k, C and D. Let S be a sample of X^d according to D with biased class noise rate β. Let S' be the instances of S that were labelled*

by C. Let D' be the distribution that is proportional to D except that it assigns zero probability to points on the boundary of C. If assumption (1) holds, then

$$\mathsf{err}(k\text{-NN}(S), C, D') \;=$$
$$1 \;-\; f \cdot \sum_{k'=\lceil \frac{k}{2} \rceil}^{k} \binom{k}{k'} B(1-\beta(1-\mu), k', k) \;-\; (1-f) \cdot \sum_{k'=\lceil \frac{k}{2} \rceil}^{k} \binom{k}{k'} B(1-\beta\mu, k', k),$$

where f is the fraction of X^d labelled 1 by C.

In the absence of additional information, we can not conclude anything about $\mathsf{err}(k\text{-NN}(S), C, D)$ (which is what one can measure empirically) from $\mathsf{err}(k\text{-NN}(S), C, D')$ (the model's prediction) or from $\mathsf{err}(k\text{-NN}(S'), C, D') = 0$ (the assumption underlying the model). For example, if D just so happens to assign zero probability to points on C's boundary, then

$$\mathsf{err}(k\text{-NN}(S'), C, D) = \mathsf{err}(k\text{-NN}(S'), C, D')$$

and so in the best case $\mathsf{err}(k\text{-NN}(S), C, D) = 0$. On the other hand, if all of D's mass is on C's boundary then in the worst case $\mathsf{err}(k\text{-NN}(S), C, D) = 1$. Furthermore, it is generally impossible to know whether $\mathsf{err}(k\text{-NN}(S'), C, D') = 0$.

Despite these difficulties, in the next section we will evaluate the model on real-world data by simply assuming $\mathsf{err}(k\text{-NN}(S'), C, D') = 0$ and $D' = D$, and comparing the predicted and observed error.

3.3 Accuracy Evaluation

We evaluated the two models against two real-world learning tasks:

- The MUSHROOM data-set from the UCI repository contains 8124 instances with 23 attributes, with no missing values.
- The PTV collaborative recommendation data for television listing [20] contains 2344 instances (people) and 8199 attributes (television programs), and only 0.3% of the matrix entries are non-null. We discarded people who rated fewer than 0.05% of the programs, and programs rated by fewer than 0.05% of the people. The resulting 241 people and 570 programs had a sparseness of 15.5%. The original ratings (values from 1–4) were converted into binary attributes ('like'/'dislike').

We used the standard k-NN algorithm with no attribute or vote weighting. Distance was measured using the Euclidean metric (ignoring non-null attributes). All experiments use $k = 10$.

Our experiments use a variation on the standard cross validation approach. We repeat the following process many times. First, we randomly partition the entire set of instances into a *real* set R, a *fake* set F, and a testing set T. To implement the biased class noise model, a *noisy* set N containing $\beta|R|/(1-\beta)$ instances is then randomly drawn from F. The class attributes of the instances in N are then modified to 1 with probability μ and 0 with probability $1-\mu$. The k-NN learning algorithm is then training on $R \cup N$ (so the noise rate is $|N|/|R \cup N| = \beta$). We measure accuracy as the fraction of correct predictions for the test instances in T. For MUSHROOM, noise is added only to the class attribute defined by the data-set's authors. For PTV the class attribute (i.e., program to attack) is selected randomly.

Absolute accuracy model. The absolute accuracy model predicts

$$\Pr[\mathsf{err}(k\text{-NN}(S_\beta), C, D) < \epsilon],$$

the probability that the accuracy exceeds $1-\epsilon$, where S_β is a sample with biased class noise rate β. Let the model's predicted absolute accuracy from Theorem 1 be

$$A_{\text{abs}}(\beta) = \Upsilon_{\text{den}}(\lceil k/2 \rceil, \epsilon/4LB, \epsilon/4, (1-\lambda)|S_\beta|) \cdot \Upsilon_{\text{spa}}(\lfloor k/2 \rfloor, \epsilon/4LB, \epsilon/4, \lambda|S_\beta|).$$

Our empirical estimate $\widehat{A}_{\text{abs}}(\beta)$ of this probability is simply the fraction of trials for which ϵ exceeds the error.

Recall that Theorem 1 requires a bound L on the perimeter of the target concept, and a bound B on the probability of any instance under the distribution D. Thus our model is not completely general, and furthermore it is difficult to estimate these parameters for a given learning task. However, it is easily shown that for small values of k, $A_{\text{abs}}(\beta)$ does not depend on L and B, and thus we do not need to tune these parameters of our model for each learning task.

Due to the worst-case analysis, typically $A_{\text{abs}}(\beta) \gg 1$ - clearly an absurd value. However, for the purposes of analysing robustness, such values are useful, because we are interested in the increase in error at noise rate β compared to $\beta = 0$. We therefore report results using the ratios $(L - A_{\text{abs}}(\beta))/(L - A_{\text{abs}}(0))$ and $(L - \widehat{A}_{\text{abs}}(\beta))/(L - \widehat{A}_{\text{abs}}(0))$, where $L = A_{\text{abs}}(1)$ is a constant chosen to scale the ratios to [0,1].

The results for MUSHROOM with $\epsilon = 0.25$ are shown in Fig. 2. The predicted and observed accuracies agree reasonably well, even accounting for the fact that the data have been scaled to [0,1]. The fit is by no means

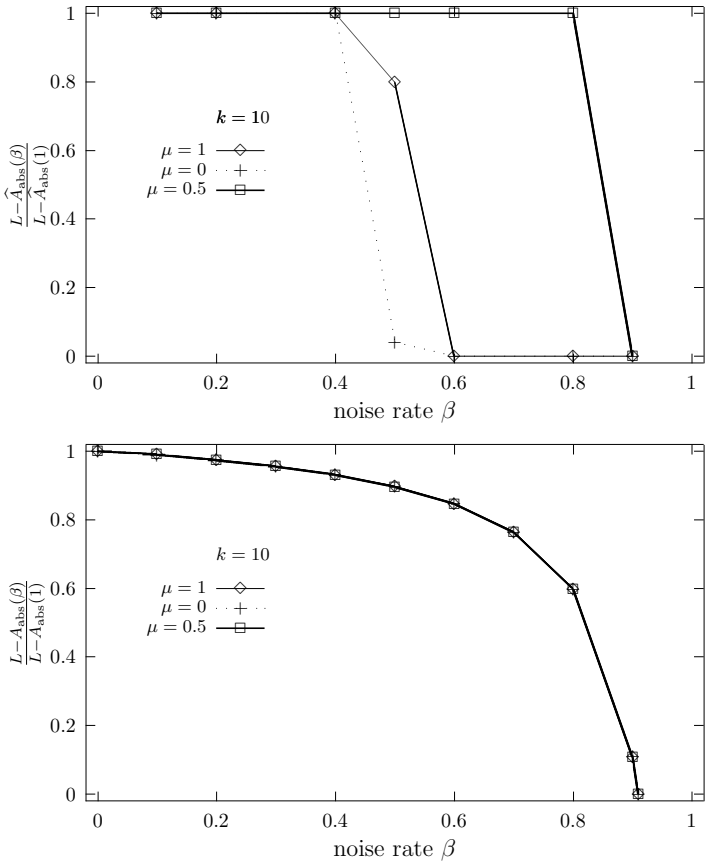

Fig. 2. MUSHROOM: Empirical (top) and predicted (bottom) scaled absolute accuracy.

perfect, but we are satisfied with these results, since worst-case PAC-like analyses are usually so weak as to be incomparable to real data.

Fig. 3 shows the results for PTV with $\epsilon = 0.3$. Here the fit is worse: PTV appears to be much more robust in practise than predicted, particular as β increases. We conjecture that this is due to the fact that the PTV data is highly noisy, but further analysis is needed to explain these data.

Relative accuracy model. The relative accuracy model predicts the error $\mathsf{err}(k\text{-NN}(S_\beta), C, D)$, where as before S_β is a sample with biased class noise rate β. Let $A_{\mathrm{rel}}(\beta)$ be the model's prediction from Theorem 2. Our

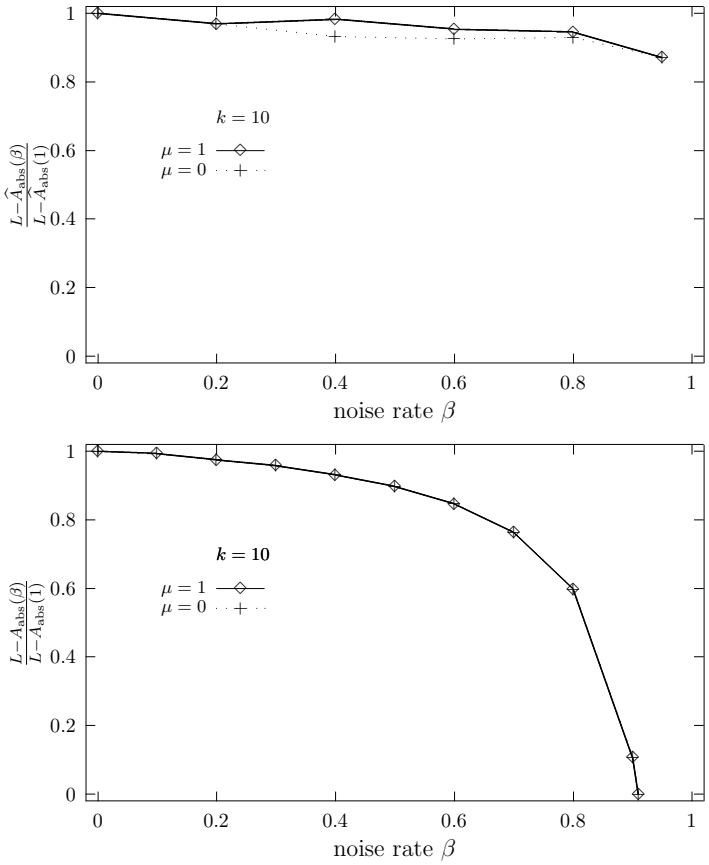

Fig. 3. PTV: Empirical (top) and predicted (bottom) scaled absolute accuracy.

empirical estimate $\widehat{A}_{\text{rel}}(\beta)$ of this probability is simply the fraction of incorrectly classified test instances. As before, we scale all data to [0-1].

The results for MUSHROOM are shown in Fig. 4 and the PTV results are shown in Fig. 5. The model fits the observed data quite well in both domains, though as before PTV appears to be inherently noisier than MUSHROOM.

4 Stability Analysis

Before we begin our analysis, it is worthwhile to review related work carried out in the area of knowledge-based systems (KBS). One of the underlying claims of KBS is that they can deal with incomplete, inaccurate or uncertain data. However, few researchers have attempted to

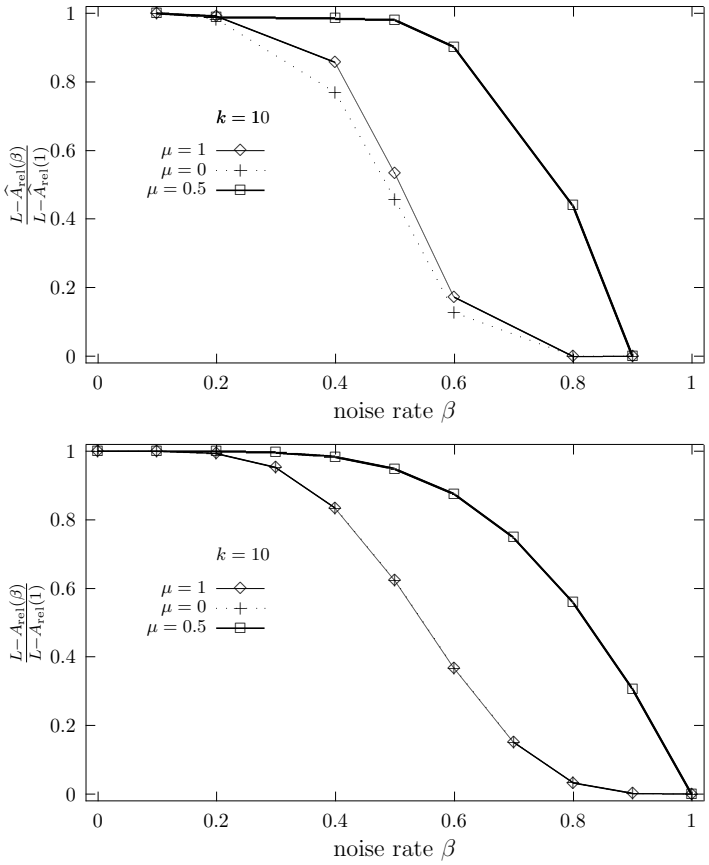

Fig. 4. MUSHROOM: Empirical (top) and predicted (bottom) scaled relative accuracy.

quantitatively analyse the degree to which this claim is true. To measure the performance of the system, the database upon which predictions are based, is typically divided into training and test sets and the predictive accuracy on the test data is measured. While this gives an indication of how well the system performs given the training data, it does not measure how the system's performance evolves as the database changes. As well as a measure of *accuracy*, a measure of *robustness* is required. In this regard, the previous work of Hsu and Knoblock [10, 11] and Groot *et al* [7] who propose methodologies for examining the robustness of knowledge discovery and knowledge-based systems is of interest.

Groot *et al* uses as a starting point the informal definition of robustness found in [12] :

The degree to which a system or component can function correctly in the presence of invalid inputs or stressful environmental conditions.

They go on to argue that robustness must be analysed through a degradation study in which the quality of the KBS output is measured as the quality of the input is gradually decreased. We note here that the robustness depends on the quality measure that is used. Groot *et al* assume a set of discrete outputs from the KBS and adopt the *precision* and *recall* output quality measures. They do not propose a single criterion for comparing the robustness of different systems, but rather suggest a number of robustness definitions, for example that one system is more robust than another if its output quality decreases more slowly, with decreasing input quality.

Hsu and Knoblock give a more precise definition of robustness. Their approach focuses on the consistency of the rule set with the database from which it is derived. They argue, intuitively, that the robustness of a rule, r, in the KBS can be measured as the probability that the database is in a state which is consistent with r. As a more tractable definition they give the following:

Given a database state d and a rule r that is consistent with d, let t denote the transactions on d that result in new database states inconsistent with r. The robustness of r in the database state d is $\text{Robust}(r|d) = 1 - \Pr(t|d)$.

Hence, the robustness of a rule given a particular database state depends on the probability that a transaction that moves it into an inconsistent state will occur. However, in their analysis, Hsu and Knoblock consider only *normal* transactions and hence examine the system's robustness only to natural noise in the data. We argue that, for open systems, in which the database is updated through system use, it is necessary also to consider how much effort is required by external users to deliberately change the system's output. Hence, it is not appropriate to measure the probability that the transaction will occur - rather we prefer to examine the *cost* of such a transaction. If a transaction that renders the system inconsistent is inexpensive to perform, then this presents a major threat to the integrity of the system.

4.1 Stability of Prediction

We take the following approach to stability of prediction (SOP). Let U be the set of all users in the system. An *attack* is a transformation T which

maps the system database D_U to a new database $D'_{U'}$. Under the transformation, each vote v in D_U is mapped to a new vote v'. A transformation may also entail the addition of new users, so that $U \subseteq U'$.

Let $\mathcal{T}$ be the set of all possible attacks. Define a cost function $C : \mathcal{T} \to \mathcal{R}$. In general the cost of a transformation is application dependent. Various criteria can be used to construct a cost function. If, for example, the cost is related to the amount of effort it takes to perform the transformation, then this could be modelled by an increasing function of the number of user-item pairs which are modified by the transformation. There may also be a real cost, if ratings can only be entered into the system by purchasing the item in question.

Fix a set, A, of user-item pairs in the database and define stability for this set. Assuming that the set of vote values is bounded above by R_{max} and below by R_{min}, for each user-item pair $(a, j) \in A$, the normalised absolute error, NAE, of prediction pre- and post-attack T is given by

$$\mathrm{NAE}(a, j, T) = \frac{|p_{a,j} - p'_{a,j}|}{\max\{|p_{a,j} - R_{max}|, |p_{a,j} - R_{min}|\}} \ . \tag{2}$$

The stability of prediction (SOP) to attacks of cost c is given by

$$\mathrm{SOP}(a, j, c) = 1 - \max_{\{T \in \mathcal{T} : C(T) \leq c\}} \mathrm{NAE}(a, j, T) \ . \tag{3}$$

Furthermore, the average stability of prediction of the recommendation system on the set A to attacks of cost c is given by

$$\mathrm{SOP}(A, c) = \frac{1}{|A|} \sum_{(a,j) \in A} \mathrm{SOP}(a, j, c) \tag{4}$$

The above definition is concerned with the general stability of a system. However, other metrics may be useful depending on the nature of an attack. Consider the class of targeted attacks whose goal is to force item ratings to a particular target value, R_{target}. Product push and nuke are examples of such targeted attacks. Given the set A of user-item pairs and an item j, let $A_j \subseteq A$ be those members of A who have rated j. Let $\mathcal{T}_c$ denote the set of all attacks of cost c from this class . We define *power of attack* (POA) for attack $T \in \mathcal{T}_c$ and item j by

$$\mathrm{POA}(A_j, j, T) = 1 - \frac{1}{|A_j|} \sum_{a \in A_j} \kappa_{a,j} \tag{5}$$

where $\kappa_{a,j} = 1$ iff $p'_{a,j} = R_{target}$ and 0 otherwise. We can now write the average POA for all such attacks of cost c over set A_j as

$$\mathrm{POA}(A_j, j, c) = \frac{1}{|\mathcal{T}_c|} \sum_{T \in \mathcal{T}_c} \mathrm{POA}(A_j, j, T) \tag{6}$$

Let I be the set of all items in A. Then the average POA of an attack of cost c on set A is:

$$\mathrm{POA}(A, c) = \frac{1}{|I|} \sum_{j \in I} \mathrm{POA}(A_j, j, c) \tag{7}$$

In our experimental evaluation (Section 4.3), the cost is a function of the number of attack profiles added to the database, and therefore of the noise rate β.

Note that in the above definitions no mention is made of the pre-attack predicted ratings ($p_{a,j}$). The reason for this is that an attack on any given user-item pair is deemed to be successful as long as the post-attack prediction is equal to the target rating. Even if a prediction was already at the target prior to attack, a successful attack should only force a prediction to change when necessary, i.e. when $p_{a,j} \neq R_{target}$.

In addition, these definitions do not contain any notion of *true* rating for the user-item pair. This reflects the distinction between the accuracy and stability performance measures. A system which reports the same ratings regardless of the dataset is very stable, though it is unlikely to be very accurate.

4.2 Attack Strategy

One of the most common similarity metrics used in recommendation systems is the Pearson correlation formula. From a system robustness point-of-view, there exists an important weakness in the formula that can be exploited to implement an attack. The particular weakness in question is that the correlation between users is calculated only over the items that *both* users have rated. Hence, users can correlate strongly even though they have few items in common. While this weakness has been noted from the predictive accuracy point-of-view and some modifications to the standard model have accordingly been proposed [4, 9, 1], we highlight here its implications for robustness of recommendation.

With Pearson's formula, the correlation between users who have *exactly* two items in common is always $+1$ or -1. If the attacker adopts a strategy of building false user profiles consisting of the item to be pushed

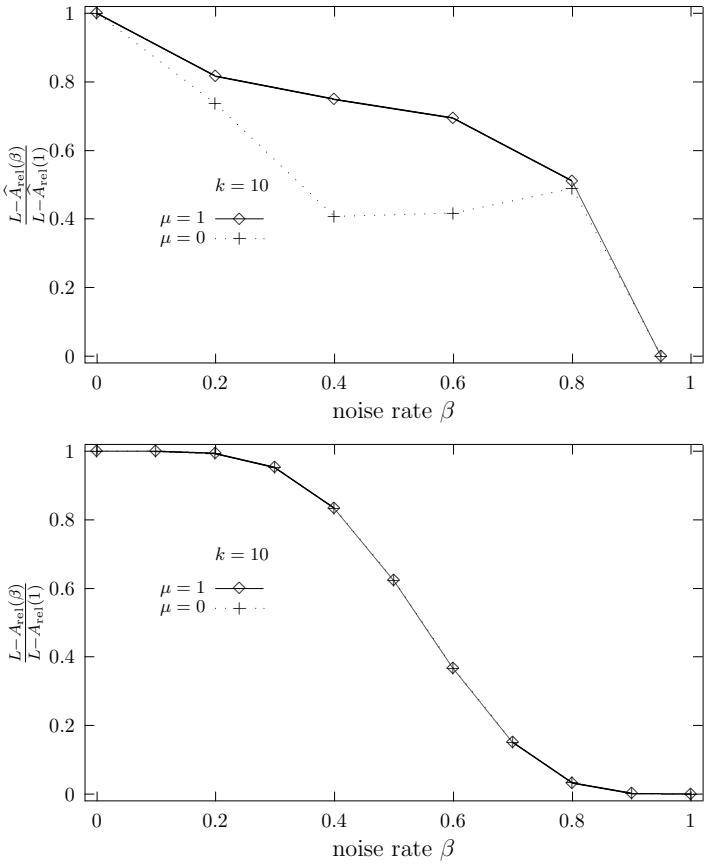

Fig. 5. PTV: Empirical (top) and predicted (bottom) scaled relative accuracy.

(or nuked) together with two other carefully selected items, then the probability of a successful attack is high. Ideally, two items about which there is strong consensus in the user population are selected to generate the attack profiles.

The remaining difficulty in this strategy is in ensuring that the attack profiles correlate in the same way (i.e. either positively or negatively) with all the target users. In practise, this means that the attacker needs to know the ratings that the target users have given to the items contained in his attack profiles. However, some simple heuristics can suffice in practise. It is a fair assumption that items that have many ratings in the database (i.e. popular items) are generally rated higher than average. Such items are therefore appropriate choices when building attack profiles.

In the following section we adopt this attack strategy and present empirical data for both product push and nuke attacks on real-world datasets.

4.3 Stability Evaluation

We used two different datasets in our evaluation:

- The MOVIELENS dataset [16] consists of 943 users, 1,682 movies and contains 100,000 transactions in total. Movies are rated on a scale of 1 to 5.
- The PTV dataset described earlier, comprising 1,542 users, 8,129 TV programs, and has 58,594 transactions. In these experiments we retain the original rating scale of 1 to 4. Note that this dataset is an order of magnitude more sparse than MOVIELENS.

From experiment we set the number of nearest neighbours to 50 for all attacks involving both datasets. An *all but one* protocol [4] is adopted in which the test set is obtained by removing a single user-item pair from the database. The standard k-NN algorithm is then used to make a prediction for this pair using all the remaining data. Potential neighbours are weighted according to the Pearson correlation metric.

We perform product push and nuke attacks on both MOVIELENS and PTV. In each attack, our strategy is as follows: a 3 item attack profile is generated consisting of the item to be pushed/nuked together with the two most popular items in the database. The identical attack profile is repeated a number of times. Note that this attack can only be successful for those users that have rated the two most popular items (35% of all users for MOVIELENS and 18% for PTV). Therefore we present stability results according to Eqn. 7 for the total set of users and only over those users who have rated the items in question.

Figs. 6 and 7 present POA against β for MOVIELENS and PTV. For MOVIELENS, both the push and nuke attacks are successful. In the case of the push attack, for example, POA falls to approximately 0.4 at $\beta = 0.06$ when averaging is done only over the users who have rated the two most popular items (labelled 'sampled' on the graph). This shows that the attack was successful in forcing 60% of all predictions to the maximum rating. This is a very significant result, given the small number (60) of attack profiles added.

For PTV, the attacks are much less successful, with POA falling only slightly below the baseline levels (i.e. the POA at $\beta = 0$, corresponding

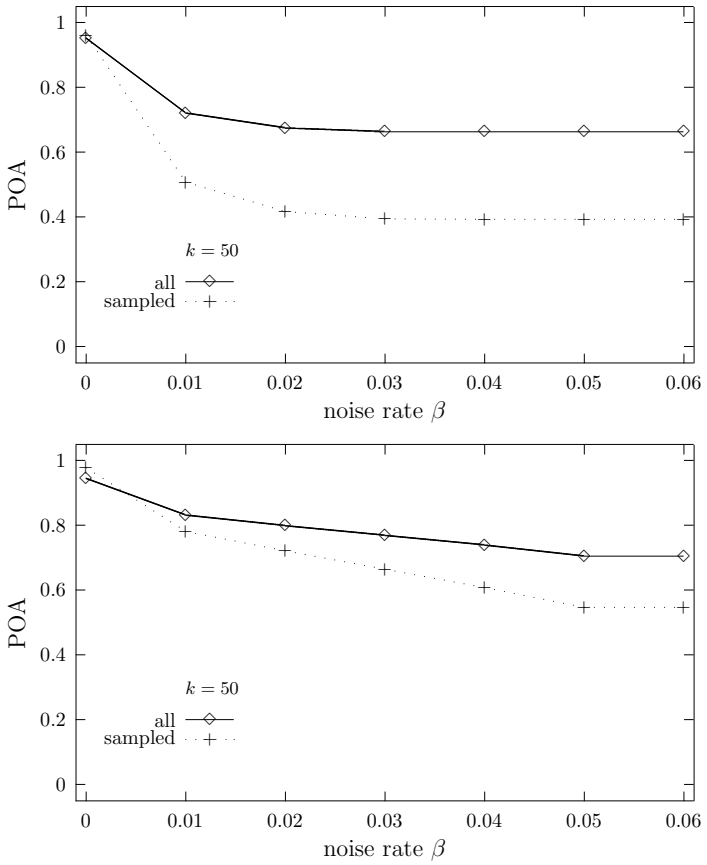

Fig. 6. MOVIELENS: POA for PUSH (top) and NUKE (bottom) attacks.

to those user-item pairs for which predictions are already at the target rating). This is explained by the fact that the average number of attack profiles that contribute to predictions is significantly less for PTV than for MOVIELENS. For example, consider the MOVIELENS database, where, at $\beta = 0.06$, 73% of all profiles that contribute to predictions are attack profiles, as opposed to only 45% for PTV (product nuke attack). Here we see the heuristic of assigning a higher rating to the most popular item in the attack profiles breaking down for PTV. This is due to the sparsity of the dataset, which is an order of magnitude more sparse than the MOVIELENS dataset.

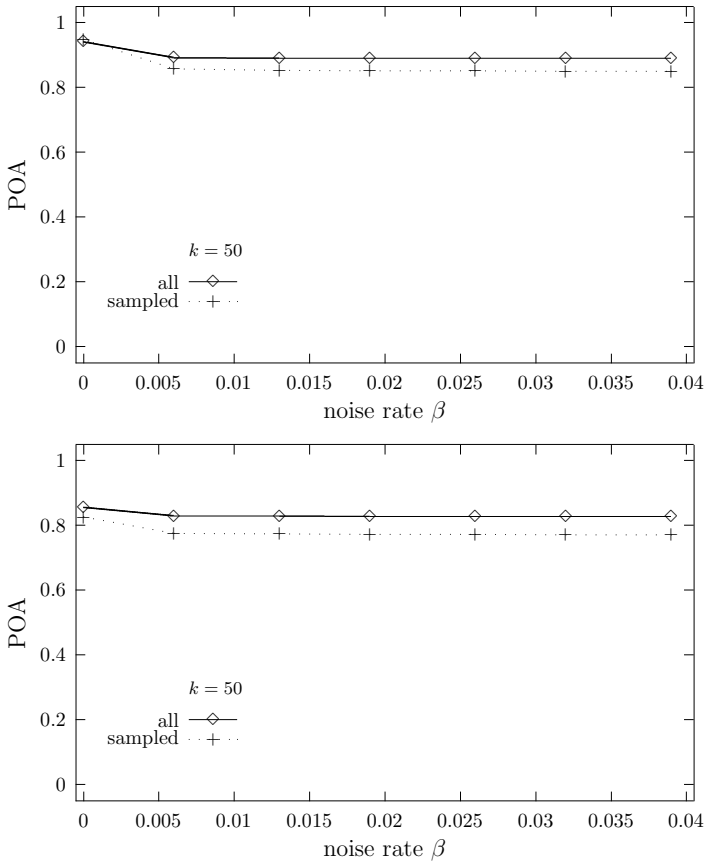

Fig. 7. PTV: POA for PUSH (top) and NUKE (bottom) attacks.

5 Related work

Collaborative recommendation has been empirically validated in numerous standard *customer/product* scenarios [4]. However, there is relatively little theoretical understanding of the conditions required for effective or accurate recommendation.

There has been substantial theoretical analysis and algorithmic work on collaborative recommendation [15, 3, 5]. For example, Azar *et al* [3] provide a uniform theoretical analysis for several information retrieval problems, including collaborative recommendation, latent semantic analysis and link-based methods such as hubs/authorities. They cast these problems as matrix reconstruction: given a matrix of objects and their attributes (e.g., for collaborative recommendation, the objects are prod-

ucts, the attributes are customers, and matrix entries store customers' ratings) from which some entries have been deleted, the task is to reconstruct the missing entries (e.g., predict whether a particular customer will like a specific product). Azar *et al* prove that the matrix entries can be efficiently recovered in as long as the original high-rank data can be mapped to some low-rank approximation, which occurs naturally in many settings.

Given the matrix-reconstruction connection between collaborative recommendation and link-based methods such as hubs/authorities [14], it would be interesting to explore the connection between our results and [17]'s investigation of the sensitivity of the linked-based techniques to perturbations to the document-document link matrix.

The fundamental difference between all of these results and ours is that the biased class noise model is more complicated than simple uniform deletion of the matrix entries. It remains an open question whether these results can be extended to accommodate our model.

6 Discussion

Collaborative recommendation has been demonstrated empirically, and has been widely adopted commercially. Unfortunately, we do not yet have a general predictive theory for when and why collaborative recommendation is effective. We have contributed to such a theory by analysing robustness, a recommender system's resilience to potentially malicious perturbations in the customer/product rating matrix.

This investigation is both practically relevant for enterprises wondering whether collaborative recommendation leaves their marketing operations open to attack, and theoretically interesting for the light it sheds on a comprehensive theory of collaborative recommendation.

We developed and evaluated two models for predicting the degradation in predictive accuracy as a function of the size of the attack and other problem parameters. The first model uses PAC-theoretic techniques to predict a bound on accuracy. This model is *absolute* in that it takes account of the exact position of the system along the learning curve, but as a worst-case model it is problematic to evaluate its predictions . In contrast, the second model makes tighter predictions, but is *relative* in the sense that it assumes perfect prediction in the absence of the malicious attack. Our preliminary evaluation of the model against two real-world data-sets demonstrates that our model fits the observed data reasonably well.

Regarding prediction stability, we have presented a framework for evaluating the stability of a system under attack. We have outlined product push and nuke attack strategies, and in our results have shown that our approach is successful (from the point-of-view of the attacker) for MOVIELENS, though not so for the more sparse dataset, PTV. While we acknowledge that certain modifications to the standard algorithm would likely reduce the effectiveness of these attacks, nevertheless we feel that the insights provided by our work are useful.

We are currently extending our work in several ways. First, our models of accuracy assume no attribute noise, which means that they cannot model the two-product attacks that exploit the weakness in the Pearson correlation metric described in Section 4.2. We are currently examining whether a general model that accounts for attribute noise is possible. Second, we are developing the same kind of rigorous predictive models for stability as have been described for accuracy. Finally, we are exploring whether the observations reported here can be leveraged to design more novel collaborative recommendation algorithms that are more robust than current techniques.

Acknowledgements. We thank Barry Smyth for the PTV data, Maírtin Keane for helpful discussion, and the Weka developers. Kushmerick was supported by grant N00014-00-1-0021 from the US Office of Naval Research, and grant SFI/01/F.1/C015 from Science Foundation Ireland.

References

1. Charu C. Aggarwal, Joel L. Wolf, Kun-Lung Wu, and Philip S. Yu. Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In *Knowledge Discovery and Data Mining*, pages 201–212, 1999.
2. M. Albert and D. Aha. Analyses of instance-based learning algorithms. In *Proc. 9th Nat. Conf. Artificial Intelligence*, 1991.
3. Y. Azar, A. Fiat, A. Karlin, F. McSherry, and J. Saia. Spectral analysis of data. In *Proc. 32nd ACM Symp. Theory of Computing*, 2001.
4. J. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Conf. Uncertainty in Artificial Intelligence*, 1998.
5. P. Drineas, I. Kerenidis, and P. Raghavan. Competetive recommender systems. In *Proc. 32nd ACM Symp. Theory of Computing*, 2002.
6. D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *C. ACM*, 35(12):61–70, 1992.
7. Perry Groot, Frank van Harmelen, and Anne ten Teije. Torture tests: a quantitative analysis for the robustness of knowledge-based systems. In *European Workshop on Knowledge Acquisition, Modelling and Management (EKAW'00)*. LNAI Springer-Verlag, October 2000.

8. D. Haussler. Probably approximately correct learning. In *National Conference on Artificial Intelligence*, pages 1101–1108, 1990.
9. J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of ACM SIGIR'99*, 1999.
10. Chun-Nan Hsu and Craig A. Knoblock. Estimating the robustness of discovered knowledge. In *Proceedings of the First International Conference on Knwledge Discovery and Data Mining*, Montreal, Canada, 1995.
11. Chun-Nan Hsu and Craig A. Knoblock. Discovering robust knowledge from dynamic closed-world data. In *Proceedings of AAAI'96*, 1996.
12. IEEE. *IEEE standard glossary of software engineering terminology.* IEEE Standard 610.12-1990, 1990.
13. M. Kearns and M. Li. Learning in the presence of malicious errors. In *Proc. ACM Symp. Theory of Computing*, 1988.
14. J. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proc. ACM SIAM Symp. Discrete Algorithms*, 1998.
15. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tompkins. Recommender systems: A probabilistic analysis. In *Proc. 39th IEEE Symp. Foundations of Computer Science*, 1998.
16. movielens.umn.edu.
17. A. Ng, A. Zheng, and M. Jordan. Link analysis, eigenvectors and stability. In *Proc. 17th Int. Joint Conf. Artificial Intelligence*, 2001.
18. U. Shardanand and P. Maes. Social information filtering: Algorithms for automating "word of mouth". In *Proc. Conf. Human Factors in Computing Systems*, 1994.
19. L. Valiant. A theory of the learnable. *Communications of the ACM*, pages 1134–1142, 1984.
20. www.changingworlds.com.

A Proof of Theorem 1

We begin with bounds on the probabilities of certain outcomes in binomial and trinomial processes whose parameters are only partially known.

Lemma 1. *Consider a binomial process where outcome o_1 has probability at least ρ and outcome o_2 consumes the remaining probability mass. The probability of exactly s o_1-outcomes in t trials is at most*

$$\Phi_2(\rho, s, t) = \binom{t}{s} B\left(\max\left\{s/t, \rho\right\}, s, t\right),$$

where

$$B(p, s, t) = p^s(1-p)^{t-s}.$$

Proof. $\binom{t}{s}B(p,s,t)$ is the probability of s successes in t trials of a binomial process with success probability p. We need an upper-bound on this probability when we know only that $p \geq \rho$. $\frac{\partial B}{\partial p} = 0$ at $p = \frac{s}{t}$. B increases over $0 \leq p \leq \frac{s}{t}$ and decreases over $\frac{s}{t} \leq p \leq 1$. Therefore B is maximal at $p = \frac{s}{t}$, and $\max\left\{\frac{s}{t}, \rho\right\}$ is the value of p that maximises B subject to the constraint that $p \geq \rho$. □

Lemma 2. *Consider a trinomial process where outcome o_1 has probability at least ρ and outcomes o_2 and o_3 consume the remaining probability mass. The probability of exactly r o_1-outcomes and s o_2-outcomes in t trials is at most $\Phi_3(\rho, r, s, t) = \binom{t}{r}\binom{t-r}{s} T\left(\max\left\{\frac{r}{t}, \rho\right\}, r, \frac{s}{t}, s, t\right)$, where $T(p, r, q, s, t) = p^r q^s (1-p-q)^{t-r-s}$.*

Proof. $\binom{t}{r}\binom{t-r}{s} T(p, r, q, s, t)$ is the probability of r o_1-outcomes and s o_2-outcomes in t trials of a trinomial process where o_1 has probability p and o_2 has probability q. We need an upper-bound on this probability when we know only that $p \geq \rho$. $\frac{\partial T}{\partial p} = 0$ at $p = \frac{r}{t}$, and $\frac{\partial T}{\partial q} = 0$ at $q = \frac{s}{t}$. T increases over $0 \leq p \leq \frac{r}{t}$ and $0 \leq q \leq \frac{s}{t}$, and decreases over $\frac{r}{t} \leq p \leq 1$ and $\frac{s}{t} \leq q \leq 1$. Therefore T is maximal at $p = \frac{r}{t}$ and $q = \frac{s}{t}$, and $\max\left\{\frac{r}{t}, \rho\right\}$ is the value of p and $\frac{s}{t}$ is the value of q that maximise T subject to the constraint that $p \geq \rho$. □

The following lemma bounds the probability that a subset is sparse or dense.

Lemma 3. *The following holds for any d, α, γ, k, and distribution D. Let $m = \lceil \frac{\sqrt{d}}{\alpha} \rceil$ and $\rho = \frac{\gamma}{m^d}$. The probability that a sample S of X^d drawn according to D is (k, α, γ)-dense is at least*

$$\Upsilon_{\mathrm{d}}(k, \alpha, \gamma, |S|) \;=\; 1 - m^d \sum_{0 \leq k' < k} \Phi_2(\rho, k', |S|), \tag{8}$$

and the probability that S is (k, α, γ)-sparse is at least

$$\Upsilon_{\mathrm{s}}(k, \alpha, \gamma, |S|) \;=\; 1 - m^d \sum_{\langle k', k'' \rangle \in \mathcal{K}} \Phi_3(\rho, k', k'', |S|), \tag{9}$$

where $\mathcal{K} = \{\langle k', k'' \rangle \mid 0 \leq k', k'' \leq |S| \wedge k < k' + k'' \leq |S|\}$.

Proof. First consider (8). Partition X^d into m^d squares, where m is chosen large enough so that any two points in one square are at most distance α apart. By the Pythagorean Theorem, we require that $m \geq \frac{\sqrt{d}}{\alpha}$, so choose $m = \lceil \frac{\sqrt{d}}{\alpha} \rceil$. Let F be the set of *frequent* squares: those with probability at least $\rho = \frac{\gamma}{m^d}$. Since there are at most m^d squares not in F, the total probability of the non-frequent squares is at most $m^d \rho = m^d \frac{\gamma}{m^d} = \gamma$. If at least k sample points lie in each of the frequent squares, then the sample will be sufficiently dense for the points in the frequent squares. The probability of selecting a point in some particular heavy square is at least ρ, and the probability of not selecting a point in this square is

at most $1 - \rho$. By Lemma 1, the probability of selecting exactly k' out of $|S|$ points in some particular heavy square is at most $\Phi_2(\rho, k', |S|)$. Therefore the probability of selecting fewer than k points in some particular heavy square is at most $\sum_{0 \leq k' < k} \Phi_2(\rho, k', |S|)$. This expression is an upper bound on the probability that the sample is insufficiently dense for some particular frequent square. Since there are at most m^d frequent squares, the probability that the sample is insufficiently dense for one or more frequent squares is at most $m^d \sum_{0 \leq k' < k} \Phi_2(\rho, k', |S|)$. (8) follows by noting that the probability that the sample is sufficiently dense for every frequent square is therefore at least $1 - m^d \sum_{0 \leq k' < k} \Phi_2(\rho, k', |S|)$.

We now prove (9). As before, partition X^d into m^d squares. The *border* of a square is the set of points outside but within α of the square. If at most k sample points lie in each of the frequent squares and their borders, then the sample will be sufficiently sparse for the points in the frequent squares. Consider some frequent square in the set F of squares with probability at least $\rho = \frac{\gamma}{m^d}$. By Lemma 2, we have that $\Phi_3(\rho, k', k'', |S|)$ is an upper bound on the probability that k' sample points fall in some specific heavy square and k'' points fall in its border. Therefore the probability of selecting more than k points in some particular frequent square or its border is at most $\sum_{k', k''} \Phi_3(\rho, k', k'', |S|)$, where the sum is over the combinations of k' and k'' satisfying $0 \leq k', k'' \leq |S|$ and $k < k' + k'' \leq |S|$. Since there are at most m^d frequent points, (9) follows by noting that the probability that the sample is sufficiently dense for every frequent point is at least one minus m^d times this expression. □

Finally, our proof of Theorem 1 derives from [2]; here we sketch the main ideas.

Let $C \in \mathcal{C}_L$ be the target concept. Let $C_\alpha^\subseteq \subset C$ be the *core* of the concept: the points at least a distance α away from some point not in C. Let $C_\alpha^\supseteq \supset C$ be the concept's *neighbourhood*: the points within a distance α of some point in C. As introduced in Section 3.1, let S_{good} be the noise-free instances in a sample S and $S_{\text{bad}} = S \setminus S_{\text{good}}$ be the noisy instances. Suppose that S_{good} is $\langle \lceil k/2 \rceil, \alpha, \gamma \rangle$-dense and S_{bad} is $\langle \lfloor k/2 \rfloor, \alpha, \gamma \rangle$-sparse. Let k-NN(S) be the set of points labelled by k-NN given training data S. Then, except for a set U of instances with probability at most 2γ, k-NN(S) is bounded within a distance α of the target concept: $(C_\alpha^\subseteq \setminus U) \subseteq (k\text{-NN}(S) \setminus U) \subseteq (C_\alpha^\supseteq \setminus U)$.

We can therefore bound the error of k-NN(S) by bounding the probability of $C_\alpha^\supseteq \setminus C_\alpha^\subseteq$. The perimeter of C is at most L, and therefore the volume of $C_\alpha^\supseteq \setminus C_\alpha^\subseteq$ is at most $2\alpha L$. No point in X^d has probability greater than B, and therefore the probability of $C_\alpha^\supseteq \setminus C_\alpha^\subseteq$ is at most $2\alpha LB$.

Therefore, assuming that S_{good} is $\langle \lceil k/2 \rceil, \alpha, \gamma \rangle$-dense and S_{bad} is $\langle \lfloor k/2 \rfloor, \alpha, \gamma \rangle$-sparse, we have that $\mathsf{err}(k\text{-NN}(S), C, D) < 2\gamma + 2\alpha LB$. The first term counts instances whose noisy neighbours outvoted noise-free neighbours, and the second term counts mistakes that might occur on the boundary of C.

To complete the proof, we must ensure that $2\gamma + 2\alpha LB < \epsilon$. Since we seek a lower bound on the probability that $\mathsf{err}(k\text{-NN}(S), C, D) < \epsilon$, we can simply split the total permissible error equally between the two causes: $2\gamma = \epsilon/2$ and $2\alpha LB = \epsilon/2$, or $\gamma = \epsilon/4$ and $\alpha = \epsilon/4LB$.