Inflow, Outflow, and Reciprocity in Machine Learning

Mukund Sundararajan [1] **Walid Krichene** [2]

Abstract

Data is pooled across entities (individuals or enterprises) to create machine learning models, and sometimes, the entities that contribute the data also benefit from the models. Consider for instance a recommender system (e.g. Spotify, Instagram or YouTube), a health care app that predicts the risk for some disease, or a service built by pooling data across enterprises.

In this work we propose a framework to study this value exchange, i.e., we model and measure contributions (**outflows**), benefits (**inflows**) and the balance between contributions and benefits (the degree of **reciprocity**). We show theoretically, and via experiments that under certain distributional assumptions, some classes of models are approximately reciprocal.

These results only scratch the surface; we conclude with several open directions.

1. Introduction

Machine learning (henceforth ML) depends on training data, and as machine learning and the artificial intelligence powered by it have increased in importance, there is an interest in understanding the role of data. This data may be collected from raters who are compensated for labeling the data, or from end-users of products powered by ML, or from enterprises that own and license data.

Different approaches have been considered to study the role of data. Techniques such as (Jia et al., 2019; Ghorbani & Zou, 2019; Hara et al., 2019; Pruthi et al., 2020; Hoaglin & Welsch, 1978; Koh & Liang, 2017a; Yeh et al., 2018) measure the contribution of data to model quality. Data has also been viewed through the lens of privacy (for instance, (Dwork & Roth, 2014)), which asks whether an individual's contributions reveal something about the individual.

However, notice that both of these views are one-sided: they are concerned with the *contributions* of the individual to the system, but do not take into consideration how much the individual *benefits* from the system. In this work, we study contributions, benefits, and the balance between them.

1.1. Running Examples

In some applications, the agents that contribute data to a system also *benefit* from it. We would like to study this *inflow* of value to the user, and the resulting exchange of value stemming from contributions and benefits.

Recommender Systems: Popular recommender systems such as Spotify, Instagram or YouTube recommend items (e.g. music, videos, posts) to their users. Past interactions of an individual with items are used to learn not only the preferences of the individual, but also the characteristics of the item, that are then used to recommend it to other individuals. This is the premise of collaborative filtering (Koren & Bell, 2015). Indeed, an individual's data helps others and vice-versa. Furthermore, this applies to any system that leverages user behavior (such as clicks on content, or the length of engagement with the content) to determine recommendations or rankings.

Healthcare: Alternatively, consider initiatives that collect health data from several individuals and use this to build models that predict sleep, or the risk of disease (Perez et al., 2019; Gulshan et al., 2016; Wang et al., 2016; Brajer et al., 2020; Rajkomar et al., 2018). Such models benefit the individual contributing the data, besides their data benefiting other individuals that use the model's predictions.

Federated Learning: Federated Learning (FL) is a distributed training techniques where individuals (or larger entities such as enterprises) build shared models in a decentralized way, without the data moving to a central location. This is important in domains where data is scarce and pooling data from different entities can lead to significant quality improvements. One problem in federated learning is the incentives for entities to contribute data. This is particularly true in the cross-silo FL setting, in which larger entities (such as companies) contribute their data to train a shared model. As the survey (Kairouz et al., 2021, Section 2.2) discusses, "Clients might worry that contributing their data to training federated learning models will benefit their com-

[1] Google [2] Google Research. Correspondence to: Mukund Sundararajan <mukunds@google.com>.

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

petitors, who do not contribute as much but receive the same final model nonetheless."

1.2. Inflow, Outflow and Reciprocity

Generally, an *individual*[1] contributes data to an ML based system, and this results in a benefit to other users of the system—this is the individual's *outflow*—and they, in turn, benefit from data contributed by others—this is the individual's *inflow*.

There are various questions one can ask about these inflows and outflows. Are they large or small? Are they positive or negative? (In the recommender system example, an outflow is negative if one individual's data hurts the recommendation quality of another, perhaps because the two have opposing tastes.) Do outflows and inflows vary greatly across individuals?

While we touch upon these in our analyses, our main focus is on whether the inflows and outflows are balanced individual by individual; i.e., are they *reciprocal*.

Reciprocity is a well-studied concept in sociology (Gouldner, 1960), in economics, (e.g. Fehr & Gächter (2000)) in matching markets such as kidney or student exchanges (Gill et al., 2017; Guibert & Rayón, 2021; European Commission et al., 2020). In all of these cases, reciprocity enables systems to function based on (approximately) fair trades, without the need for money to change hands.

Indeed, we would like to understand when the value exchange from contributing data to a system is 'fair' in this sense. **We say that a system is (approximately) reciprocal if for most individuals, the individual's inflow matches their outflow.** This is formalized in Section 2.

1.3. Actionability

While the focus of this work is on measuring inflows, outflows and reciprocity, we briefly discuss some actions that may stem from such an analysis.

In the recommender system, federated learning and healthcare examples, individuals (or enterprises) with large outflows and small inflows could perhaps be compensated for their contributions, and conversely, individuals with large inflows and small outflows could pay. This is particularly true for systems in which a user can opt out from data collection entirely, but still use the system, potentially resulting in large unfairness (such individuals would have, by definition, no outflow, but still benefit from other users' data. Measuring their *inflow* quantifies this imbalance). Payments need not be in purely monetary terms. In a recommender system, users with contributions that exceed benefits could be given preferential access to new content, or fewer ads.

Another alternative is to design training algorithms that enforce a certain level of reciprocity, see discussion in Section 6.

1.4. Reciprocity as a Guardrail

Reciprocity should not be thought of as an objective, it is not something we should optimize for in isolation. It is possibly best seen as a *guardrail*, similar to privacy (Dwork & Roth, 2014) and fairness (Mehrabi et al., 2021). To elaborate on the analogy, privacy (fairness) is often used as a constraint, and the goal is to obtain the best model quality under a certain privacy budget (fairness constraint). Similarly, reciprocity can be used as (one of possibly many) constraints. We are not arguing that all ML should be reciprocal, rather that the inflows, outflows and their imbalance are worth measuring.

1.5. Our Contributions

Our primary contribution is to initiate the study of reciprocity and propose a measurement approach (Section 2). We build on top of previously proposed techniques (e.g. Pruthi et al. (2020); Hoaglin & Welsch (1978); Koh & Liang (2017a); Yeh et al. (2018)) that quantify the influence of individual training examples on individual predictions, and use these to measure contributions (outflow), benefits (inflow) and reciprocity.

Our main theoretical result (Theorem 3.2) states that models trained using Stochastic Gradient Descent, a popular training algorithm for neural networks, are strongly reciprocal under an assumption over the data distribution (Assumption 3.1). One key observation is that the influence measure satisfies certain symmetry properties.

We also demonstrate how to compute inflows and outflows efficiently, avoiding a naive quadratic complexity over the number of data points, which would make measurement infeasible (see Section 4).

Finally, we perform experiments on one recommendation and two healthcare data sets (Section 5). We observe that reciprocity is still satisfied to a large extent, even though some of the assumptions from Theorem 3.2 are violated.

2. Modeling Reciprocity

2.1. Setup and Notation

Let Z be a training data set of labelled examples. The model is learned using this data and applied to an inference set Z'.

[1] By individual, we just mean an agent that owns some data. This could also be an enterprise as in the Federated Learning example.

For an individual u, let Z_u be the set of training data examples that belong to u, similarly let Z'_u be the set of inference examples that belong to u.

Every example $z = (x, y)$ consists of input features x and a label y (the prediction target or the response variable).

2.2. Measuring Inflow, Outflow, and Reciprocity

We propose a measure of reciprocity by building on previously proposed techniques that quantify the influence of training data points on predictions (Pruthi et al., 2020; Hoaglin & Welsch, 1978; Koh & Liang, 2017a; Yeh et al., 2018). An influence technique measures the influence of a single training example z on a single inference example z', and will be denoted $\texttt{Influence}(z, z')$. We will consider two standard examples: `Marginal` influence, which generates this measure by deleting training examples, and `TracIn` influence, which tracks the effect of a training example on a prediction via the parameter changes that occur during training. We will describe these influence techniques in more detail in Section 2.3.

Consider an individual u (the protagonist). We aggregate $\texttt{Influence}(z, z')$ of the protagonist's training and inference examples as follows.

Inflow is the influence of other individuals' ($v \neq u$) data on the predictions of u, i.e.:

$$I_u = \sum_{z \in Z \setminus Z_u} \sum_{z' \in Z'_u} \texttt{Influence}(z, z'). \qquad (1)$$

Outflow is the influence of the protagonist u's training data on other individuals:

$$O_u = \sum_{z' \in Z' \setminus Z'_u} \sum_{z \in Z_u} \texttt{Influence}(z, z'). \qquad (2)$$

There is also the influence of the protagonist on their own predictions, $\sum_{z \in Z_u} \sum_{z' \in Z'_u} \texttt{Influence}(z, z')$. But since reciprocity is only concerned with flows between the protagonist and other individuals, this self-influence does not play a role in reciprocity.

We now define reciprocity for an individual, and for a population of individuals.

Definition 2.1. A machine learning model is α-**reciprocal** for individual u if the ratio of outflow to inflow I_u/O_u is in the range $[\alpha, 1/\alpha]$ for some $\alpha \in [0, 1]$. If the signs of I_u and O_u do not match, we say that the model is 0-reciprocal for individual u.

A model is (p, α)**-reciprocal for a population of individuals if it is** α**-reciprocal for** p **fraction of individuals**.

Thus, a model is at best $(1, 1)$ reciprocal and at worse $(0, 0)$ reciprocal. Depending on the measure of influence that we use, we will say (p, α)-`TracIn`-reciprocal or (p, α)-`Marginal`-reciprocal.

Another measure one can study is the correlation (e.g. Pearson or Spearman correlation) between the inflow and the outflow across individuals. One important difference between reciprocity and Pearson correlation is that the latter is sensitive to the magnitude of the flows, individuals with large flows dominate the correlation measure; while reciprocity is less sensitive to magnitudes since it is a statement about the ratios of inflows and outflows. Spearman correlation measures the correlation of ranks and is not sensitive to magnitudes, but notice that a high correlation of ranks does not imply that the inflows and outflows are *balanced*, which is what our measure seeks to capture.

Remark 2.2 (Interpreting Reciprocity). Reciprocity does not require that the outflows (or inflows) be equal or similar across individuals. Indeed, in a recommender system, a frequent user is likely to have larger outflows (and inflows) in comparison to an occasional user. Reciprocity only requires that the outflows and inflows be balanced individual-by-individual, i.e., an individual who contributes a lot, benefits a lot, and one who contributes a little benefits a little.[2]

Reciprocity does not even require that inflows and outflows be positive. Indeed, in a recommender system, the data of an individual with atypical tastes may hurt the recommendations to other individuals. Reciprocity only demands that if inflow is negative, outflow should be equally negative.

In general, reciprocity can be affected by the choice of model class, and by the data distribution. We give some examples to illustrate. Consider the following example.

Example 2.3 (k-Nearest Neighbors). In kNN, there is a distance function on points in feature space, and the prediction of the algorithm is the mean (for regression) or majority (for classification) of the k nearest neighbors, as per the distance function, of the prediction point.
Suppose there is an outlier individual (the protagonist) with outlier data points in the feature space. The protagonist will not have any influence on other individuals because the protagonist's training examples will not appear in the top k nearest list for prediction points of other individuals. However, the protagonist's predictions will be influenced by the training data of other individuals.

The distribution of interactions matters. If all of an individual's interactions with the system are late in the system's lifetime, this individual will benefit but not contribute. In the analysis, we will make a stationarity assumption to control for this effect, see Assumption 3.1.

[2] Contrast this with differential privacy (Dwork & Roth, 2014), where the goal is asymmetric: one seeks to limit the impact of a user's data on the model (for example via gradient clipping (Abadi et al., 2016)) regardless of the user's benefit from the model.

2.3. Influence Techniques

As discussed earlier, reciprocity is parameterized by a technique that identifies the *influence* of a training example on a prediction example. There are several ways to measure influence; in this paper, we investigate two methods.

2.3.1. MARGINAL INFLUENCE

Let M_Z be the machine learning model trained on the data set Z; $M_Z(x)$ is the model's prediction on a data point x. Let $\ell(\hat{y}, y)$ be a loss function measuring the loss for a prediction $\hat{y}$ and label y. The marginal influence of a training example z on an inference example $z' = (x', y')$ is based on the *counterfactual* of removing the example z from the training set:

$$\texttt{Marginal}(z, z') = \ell(M_{Z\setminus\{z\}}(x'), y') - \ell(M_Z(x'), y'). \tag{3}$$

We will adopt the convention that influence functions measure *reduction in the loss* of the inference example from the presence of a training example. Thus a positive quantity connotes that loss was reduced by that amount, and a negative quantity connotes that loss was increased by that magnitude.

Computing exact marginal influence often requires retraining the model on the modified data set. However, this can be estimated without retraining, via certain Hessian approximations (Koh & Liang, 2017b).

Remark 2.4. *Marginal* influence relies on deleting a single individual's data and retraining the model to optimality. This may result in a substantially similar model, if the data set is large. This makes this measure of influence more susceptible to noise, as will be confirmed in our experiments.

2.3.2. TRACIN INFLUENCE

The second influence measure we use is called `TracIn` (Pruthi et al., 2020). Whereas `Marginal` relies on a counterfactual approach, `TracIn` assigns contributions and benefits based on *actual* work done during the training process. It is therefore reliant on the training algorithm, and applies only to models trained using Stochastic Gradient Descent (SGD). Suppose the model M_w is parameterized by a vector $w \in \mathbb{R}^p$, and let $L_z(w) = \ell(M_w(x), y)$ be the loss of the model on an example $z = (x, y)$. In SGD, the weight parameters are updated iteratively: at each iteration t, a random batch B_t of training examples is visited, and the parameters are updated in the direction opposite (we are minimizing loss) to the gradient: $\sum_{z \in B_t} \eta_t \nabla L_z(w_t)$; here $\nabla L_z(w_t)$ is the gradient of the loss with respect to the weight parameters w_t, and η_t is the step-size at time t.

The idea of `TracIn` is as follows: First, suppose that SGD visits examples one at a time, i.e., the batch size is one. Then, the visit (to the training example z) changes the model parameters, and this changes the model's loss on the prediction example z'. It is natural to attribute this change in loss to the contribution of example z and the benefit of example z'.

If the model visits a batch B_t of training examples at once, we have to disentangle the outflows of the examples $z \in B_t$. `TracIn` does this using dot products of gradients: $-\eta_t \nabla L_z(w_t) \cdot \nabla L_{z'}(w_t)$. The term $-\eta_t \nabla L_z(w_t)$ captures the change in the weight parameters due to example z; this is by definition of gradient descent. And $\nabla L_{z'}(w_t)$ models change in the loss of the prediction example z' due to a change in the weight parameters. The influence of training example z on inference example z' is computed by summing across all the batches in which the example is present:

$$\texttt{TracIn}(z, z') = \sum_{t:\ z \in B_t} \eta_t \nabla L_z(w_t) \cdot \nabla L_{z'}(w_t). \tag{4}$$

Remark 2.5. The use of gradients entails a first-order approximation. The actual change in loss of the example z', can be written as $L_{z'}(w_{t+1}) = L_{z'}(w_t) + \nabla L_{z'}(w_t) \cdot (w_{t+1} - w_t)$, plus a higher-order term of order $O(\eta_t^2)$, that is ignored. This approximation is reasonable when the step-sizes are small. We measure this discrepancy for our experiments (see Figure 5).

3. Models Trained Using SGD Are `TracIn` Reciprocal

In this section, we study `TracIn` reciprocity for models trained using SGD. We make the following assumption for our analysis:

Assumption 3.1. Let there be a population U of individuals, a set X of features and a set Y of labels. We assume that the training and inference sets are both drawn IID from a joint distribution over $U \times X \times Y$.

This is a standard assumption in machine learning literature. Notice that the assumption allows for different individuals to have different distributions (individuals are *not interchangeable*). What the assumption requires is that for a given individual, the training and inference distributions be the same, i.e. that an example is equally likely to be in the training or the inference set. We discuss breakages of this assumption further in Section 6.

Theorem 3.2. *Consider a model trained with Stochastic Gradient Descent for T steps. Suppose that Assumption 3.1 holds. Furthermore, suppose that batches of training data $(B_t)_{t \in \{1,\dots,T\}}$ are mutually independent. Then the model is $(1,1)$-`TracIn`-reciprocal in expectation, in the sense that for all individuals u, $\mathbb{E}[I_u] = \mathbb{E}[O_u]$.*

We give a sketch of the argument, the full proof is deferred to the appendix. Given a pair of users (u, v), the `TracIn` influence at time t of $z \in Z_u$ on $z' \in Z'_v$ is $\eta_t \nabla L_{z'}(w_t) \cdot \nabla L_z(w_t)$, which is symmetric in z, z'. Conditioned on the model parameters at time step t, Z_u, Z'_u have the same distribution, and Z_v, Z'_v have the same distribution. This allows us to argue that the expected influence of u on v at time t is the same as the expected influence of v on u. The proof formalizes this argument.

Remark 3.3. The theorem suggests that reciprocity holds whether the inflows and outflows are positive or negative; indeed an individual's data could hurt another's predictions if their characteristics are very different. (In our model, negative outflows and inflows would manifest as negative dot-products $\nabla L_z(w_t) \cdot \nabla L_{z'}(w_t)$, what helps one individual hurts the other.) It holds irrespective of the variation in inflows and outflows across individuals; indeed some individuals may contribute more data than others (depending on the size of Z_u).

Remark 3.4. The theorem assumes that at each step of gradient descent, a new set of independent examples is drawn, which precludes revisiting the same example multiple times. In practice, training examples are revisited, which breaks independence. But notice that for the result to hold, it suffices that future samples B_t be independent of the past trajectory $w_0, \dots, w_{t-1}$. In some regimes, this may be a reasonable approximation. For example, when the batch size is very large, there is little variance in the gradients, and one can informally treat the trajectory $w_0, \dots, w_{t-1}$ as being deterministic, unaffected by random sampling. Another regime is when the data set is very large, and training only requires a very small number of passes over the training data. In such cases, independence may be a reasonable approximation. The experiments in Section 5 suggest that models can be approximately reciprocal even when the independence assumption is broken.

Remark 3.5. The proof crucially relies on the symmetry of the influence function $\eta_t \nabla L_{z'}(w_t) \cdot \nabla L_z(w_t)$ in z, z'. Thus, any modifications to SGD that break this symmetry may also break reciprocity. In particular, it is common to clip (rescale) gradients to enforce differential privacy (Abadi et al., 2016). This modification breaks the symmetry because clipping affects the gradient update $\nabla L_z(w_t)$, but not the gradient of the inference example $\nabla L_{z'}(w_t)$. The purpose of clipping in this case is to control the influence of any individual on the model, it is therefore expected that reciprocity is broken. Clipping and normalization are also used for stabilizing training of deep neural networks (Pascanu et al., 2013), which may also break reciprocity.

Remark 3.6 (Homogeneity). If users are homogeneous (i.e., the marginal distributions are identical across users), then by a simple argument, any model, whether it is trained using SGD or not, would be $(1, 1)$-reciprocal. (Indeed our example of non-reciprocity (Example 2.3) based on k-Nearest Neighbors relies on outliers that differ from other users.) Theorem 3.2 allows for heterogeneity among users.

4. Efficient Computation of `TracIn` Flows

The definition of Inflows and Outflows as a sum of terms of the form $\texttt{Influence}(z, z')$ may suggest that one needs to compute this matrix of pairwise influence, which can be prohibitively expensive, but we show that for `TracIn`, this can be done more efficiently. This optimization was necessary in our experiments.

Suppose we apply minibatch SGD, and let B_t be the batch at step t. A naive computation of outflow (2) would suggest that we need to compute, at each step t, $\texttt{Influence}(z, z')$ for all $z \in B_t$ and $z' \in Z'$. This represents $|B_t||Z'|$ dot products, and if p is the dimension of the parameter space, the total cost per step t would be $O(|B_t||Z'|p)$.

Observe that by equations (2) and (4), we can write the outflow as

$$\begin{aligned} O_u &= \sum_{z \in Z_u} \sum_{z' \in Z' \setminus Z'_u} \sum_{t: z \in B_t} \eta_t \nabla L_{z'}(w_t) \cdot \nabla L_z(w_t) \\ &= \sum_{z \in Z_u} \sum_{t: z \in B_t} \eta_t \nabla L_z(w_t) \cdot \sum_{z' \in Z' \setminus Z'_u} \nabla L_{z'}(w_t) \\ &= \sum_{z \in Z_u} \sum_{t: z \in B_t} \eta_t \nabla L_z(w_t) \cdot (\nabla L_{Z'}(w_t) - \nabla L_{Z'_u}(w_t)), \end{aligned} \tag{5}$$

where, in the last equality, we define, for any subset $S \subseteq Z'$,

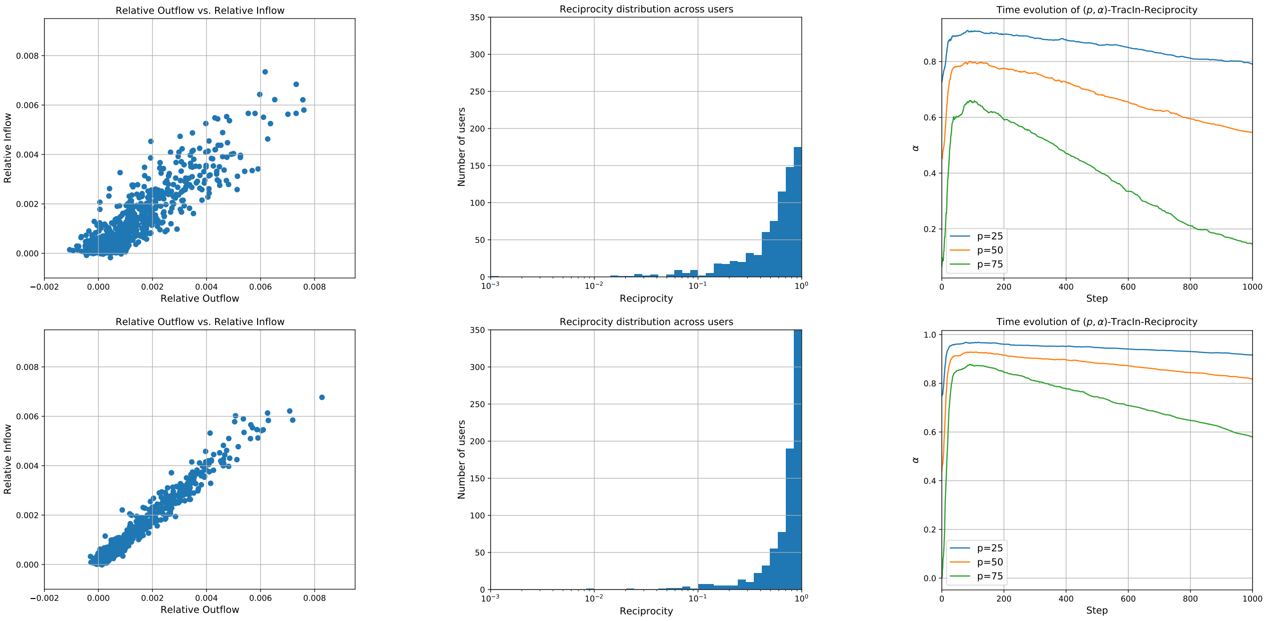
$$\nabla L_S(w_t) = \sum_{z' \in S} \nabla L_{z'}(w_t). \tag{6}$$

To compute the outflows of all individuals, from equation (5), it suffices to compute at each step t the following quantities: $\nabla L_z(w_t), z \in B_t$ (cost $O(p|B_t|)$), $\nabla L_{Z'}(w_t)$ (cost $O(p|Z'|)$), $\nabla L_{Z'_u}, u \in U$ (cost $O(p|Z'|)$), then compute one dot product for each pair $(z, u) \in B_t \times U$ (cost $O(p|B_t||U|)$). The total cost is therefore $O(p(|B_t||U| + |Z'|))$. In summary, the per-step cost is reduced from $O(p|B_t||Z'|)$ to $O(p(|B_t||U| + |Z'|))$. This is an improvement by a factor $\min(|B_t|, |Z'|/|U|)$, leading to a significant improvement when the average number of inference examples per individual is large.

5. Experiments

We perform experiments on a recommender system data set and two health data sets. Here is what we hope to learn from the experiments:

- Theorem 3.2 assumes that data points are visited exactly once during training. Practically, for many use cases, training data are visited more than once. Does reciprocity hold in such scenarios? We will find that it does, though approximately.



(a) Average outflow vs. inflow for all individuals.

(b) Histogram of reciprocity across individuals.

(c) Time evolution of (p, α)-TracIn-Reciprocity along the optimization trajectory.

Figure 1. Experiment results on the MovieLens data set. The top row shows averaged results over ten training runs for a single split, while the bottom rows shows averages across all runs and all splits (ten runs per split for ten splits).

- The theorem is a statement about inflows and outflows *in expectation*. To what extent is there reciprocity for a *single realization* of the data? We will show that in the recommender system example, there is approximate reciprocity for a single realization, though the degree of reciprocity increases as we average over realizations.

- Can `Marginal` reciprocity be measured reliably (see Remark 2.4)? Is there agreement between `Marginal` and `TracIn` reciprocity? We will find that `Marginal` reciprocity is noisy for the recommender system example, possibly because the data set is large and an individual has relatively small inflows and outflows. For healthcare datasets, we find that `Marginal` reciprocity is more stable, and there is a directional agreement between the two measures.

For each experiment, we randomly partition the data into training and inference sets; this partitioning reflects Assumption 3.1. Measurements are averaged across several such random splits.

5.1. Recommendation Data Set

We conduct experiments on MovieLens Data (Harper & Konstan, 2015), specifically, the MovieLens 100K data set with 943 individuals, 1682 items (movies), and 100,000 ratings, i.e., an average of about 106 ratings per individual. Each individual has at least 20 ratings. Each movie is rated on a scale from 1-5. We randomly split the ratings into training and inference sets in the ratio 80:20.[3]

We run the experiment on 10 different splits. For each split, we average the measurements across 10 random initializations. We measure inflows, outflows, and (p, α) reciprocity. We also report in Appendix C additional measurements, such as the first-order approximation error in `TracIn` (recall Remark 2.5), and the signal-to-noise ratio (SNR) of each measure (Appendix C.2). We find that the SNR is much lower for `Marginal` influence than for `TracIn` influence, due to the reasons discussed in Remark 2.4. For this reason, we only report results for `TracIn`.

[3] We train a matrix factorization model with embedding dimension $d = 16$. We randomly initialize the user and item embeddings. Given the relatively small size of the training data, we use full-batch Gradient Descent, i.e., all examples are visited at every time step. We use the following hyper-parameters, which we tuned on a random split of the data: regularization coefficient $\lambda = 1$, number of steps $T = 1000$, and learning rate $\eta = 0.0002$. As a sanity check, the quality of the model is consistent with previously reported results. For example, (Zhang et al., 2017; Rashed et al., 2019) report an RMSE of 0.911 using a 90-10 split. Our model has an RMSE of 0.910 on the same split.

Degree of reciprocity We plot inflows vs. outflows in Figure 1a. Inflows and outflows appear commensurate; large (resp. small) inflows correspond to large (resp. small) outflows. The linear trend is stronger when we average across ten splits (top vs. bottom figure).

Regarding reciprocity: For a single split, we find that 75% of the individuals have reciprocities in the range $[0.16, 1]$, i.e. the model is $(0.75, 0.16)$-`TracIn`-reciprocal. The correlation between inflow and outflow is 0.89.

When averaging across ten splits, **the model is (0.75, 0.58)-`TracIn`-reciprocal**, and the correlation between inflows and outflows increases to 0.98.

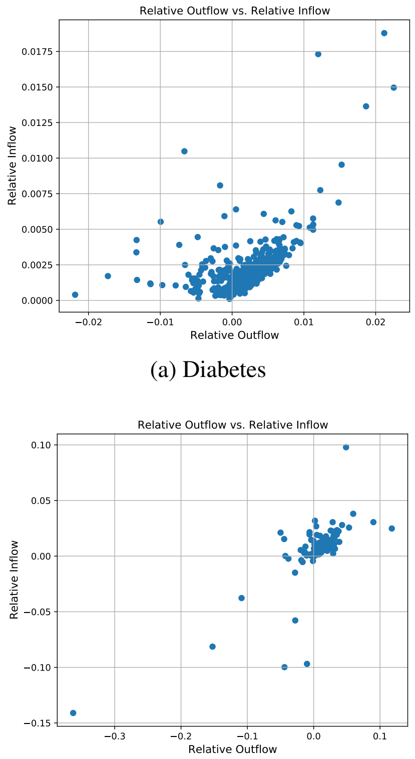
The experiment shows that although the independence assumptions of Theorem 3.2 don't hold (since training examples are revisited), reciprocity is relatively high in practice, both on a single split, and when averaged across splits. It is remarkable that we measure some degree of reciprocity even on a single split (recall that the theorem is a statement in expectation). Averaging results across splits gives an empirical estimate of the expected inflows/outflows, and indeed this averaging increases the measured reciprocity, in line with the theorem.

Signs of inflows and outflows We also observe that inflows and outflows are largely non-negative. In a single split, 0.7% of individuals have a negative inflow and 19.6% have a negative outflow. When averaging across all splits, 0.1% have a negative inflow and 8% have a negative outflow; on average, these individuals' training examples degrade the prediction quality of other individuals. However, these individuals tend to have small magnitudes of outflow and inflow; this also explains why the correlation measure (which is dominated by flows of large magnitudes) indicates a higher degree of reciprocity than the (p, α) measure.

Effect of training dynamics Figure 1c shows the time evolution of TracIn-Reciprocity along the optimization trajectory. At initialization, reciprocity is low. It quickly increases during the early phase of training, and as the model nears convergence, reciprocity starts to degrade. This may be due to cancellations in the gradients of training examples: although the aggregate gradient is small, individual training examples may have large gradients which results in large credits (inflows and outflows) being assigned.

5.2. Healthcare Data Sets

We investigate reciprocity in two healthcare data sets. The first is a data set from (Efron et al., 2004) about predicting diabetes. It has ten features: age, sex, body mass index, average blood pressure, and six blood serum measurements, and the task is to predict disease progression one year after the time of the readings. The data is from 442 individuals.



(a) Diabetes

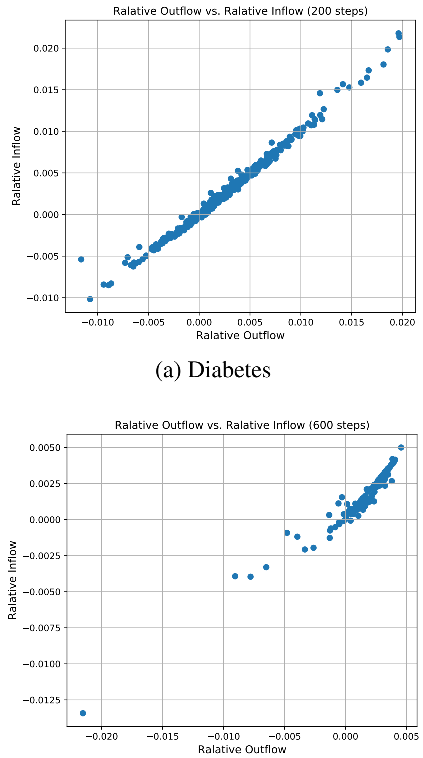
(b) Breast cancer

Figure 2. Average `Marginal`-outflow vs. `Marginal`-inflow for all individuals.

The second is a data set from the UCI Machine Learning Repository about predicting breast-cancer. There are thirty features that relate to geometric properties of cell nuclei from a digitized image of a fine needle aspirate of a breast mass. The task is to predict whether the breast cancer is malignant or benign. The data is from 569 individuals.

In both data sets, each individual corresponds to a single data point, unlike the Movielens data set where individuals correspond to at least 20 data points. Consequently, every individual belongs to exactly one of the training or inference sets; individuals in the training set only have outflows and individuals in the inference set only have inflows. In this case, measuring reciprocity is only possible in expectation over random train/test splits. More precisely, we average measurements over 100 random splits of the data.[4]

[4] On the diabetes prediction task, we train a linear regression model optimized for the mean squared error, with a number of steps $T = 200$, and a learning rate $\eta = 0.01$. On the breast cancer classification task, we train a logistic regression model with a number of steps $T = 600$ and a learning rate $\eta = 0.1$. In both cases, some features have very different scales, so we found it important to normalize all features (using mean and variance computed on the training set).



(a) Diabetes

(b) Breast cancer

Figure 3. Average `TracIn`-outflow vs. `TracIn`-inflow for all individuals.

Degree of reciprocity We compute outflows and inflows using `TracIn` and `Marginal` as measures of influence. See Figures 2 and 3. Recall from Remark 2.4 that `Marginal` is susceptible to noise. We find that the results are not as noisy as in the MovieLens experiment (the 50-th percentile of SNR is 0.30, compared to 0.19 in MovieLens), possibly because these data sets are smaller (hence individual data points have larger influence). We report results for both `Marginal` and `TracIn` so we can compare them.

We find that the diabetes model is $(0.75, 0.36)$-`Marginal`-reciprocal, and $(0.75, 0.76)$-`TracIn`-reciprocal. The breast cancer model is $(0.75, 0.48)$-`Marginal`-reciprocal and $(0.75, 0.93)$-`TracIn`-reciprocal. `TracIn` reciprocity is large for both experiments, inline with the Theorem 3.2, even though here again, the independence assumptions of the theorem don't hold. Finally, although `Marginal` reciprocity is lower, there is a directional agreement between the two measures. When one is higher, so is the other.

6. Discussion

6.1. Modeling the User

In this work, we assume that user's utility is modeled by the machine learning objective (the loss function ℓ). There are two reasons why this is not completely true. First, ML objectives are based on observable user behavior, such as the interaction with a piece of content; such observable behavior is not completely reflective of user satisfaction.

Second, this ignores any value that is created outside of the ML process. For instance in a recommender system, besides value derived from the quality of the recommendation model, there is the value created by content creators. Similarly, in a healthcare system, besides value derived from the data of other patients, some benefit stems from the healthcare professionals. The impact of this issue will vary across use-cases; for instance, when content is cheap to create, perhaps the flow through the ML is the dominant flow.

6.2. Stationarity

Our main result, Theorem 3.2, states that we have reciprocity under a certain stationarity assumption (Assumption 3.1). This assumption requires that a specific user interaction should be equally likely to occur during training or inference. In practice, this assumption will almost certainly be violated. (For instance, in a recommender system, a new user might contribute fewer examples to the training set, but have more in the inference set.)

While our theorem requires the assumption, our measurements do not. Indeed, we are able to measure inflows and outflows whatever the arrival pattern (our experiments demonstrate this), and if there is an imbalance between inflows and outflows, one can take measures to address this, via payments or different service levels.

6.3. Other Measures of Influence

Our measures of inflow and outflow depend on how we compute influence. We study two measures: `Marginal` and `TracIn`. There are other measures worth investigating, for instance, the Shapley value. Modifications of the Shapley value have been used to determine feature attribution (Sundararajan & Najmi, 2020). However, to determine the influence of training data, even approximately, would requires a number of training runs that are super-linear in the number of training examples, and this is usually intractable. That said, Shapley may be computationally feasible when the exchange happens between a small number of large entities, as in the cross-silo federated setting described in Section 1.1, or for restricted model classes such as decision trees (with additional assumptions).

6.4. Balancing Flows

One exciting open direction is the design of training algorithms that enforce a certain level of reciprocity. For example, one can imagine introducing per-example weights in SGD; changing the weight of an example z would directly scale $\texttt{TracIn}(z, z')$ for all z' (by linearity of gradients), and this can potentially be used to rebalance inflows and outflows at each training step.

This bears similarities with differential privacy (DP), which seeks to limit the impact that any training example has on the distribution of model parameters. One of the most popular methods to achieve DP is DPSGD (Abadi et al., 2016), which crucially relies on rescaling per-example gradients to bound their norm (a.k.a. gradient clipping).

Changes to the training algorithm (via example weights, sampling, or clipping) often has a non-trivial impact on model quality, and much like in DP literature, an important question will be to understand optimal trade-offs between reciprocity and model quality.

6.5. Other Settings

It is worth investigating inflow, outflow and reciprocity in any setting that involves user data, whether ML is involved or not. However, only those that have some model of user utility will be amenable to mathematical analysis. For instance, mapping applications often display real time information about traffic congestion, and the user benefits by using the application to avoid congestion. It is meaningful to measure inflow, outflow and reciprocity in such settings.

Acknowledgements

We are grateful to Amir Najmi, Kedar Dhamdhere, Steffen Rendle, Diane Tang, Fernando Pereira, John Anderson and the anonymous reviewers (from this conference and others) for insightful comments and stimulating discussions related to problem framing.

References

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pp. 308–318, New York, NY, USA, 2016. Association for Computing Machinery.

Bobadilla, J., Ortega, F., Hernando, A., and Gutiérrez, A. Recommender systems survey. *Knowledge-Based Systems*, 46:109–132, 2013.

Brajer, N., Cozzi, B., Gao, M., Nichols, M., Revoir, M., Balu, S., Futoma, J., Bae, J., Setji, N., Hernandez, A., and Sendak, M. Prospective and External Evaluation of a Machine Learning Model to Predict In-Hospital Mortality of Adults at Time of Admission. *JAMA Network Open*, 3 (2), 2020.

Dwork, C. and Roth, A. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

Efron, B., Hastie, T., Johnstone, I., and Tibshirani, R. Least angle regression. *The Annals of Statistics*, 32(2):407–451, 2004.

European Commission, Directorate-General for Education, Youth, S., and Culture. *Erasmus+ annual report 2019*. Publications Office, 2020.

Fehr, E. and Gächter, S. Fairness and retaliation: The economics of reciprocity. *Journal of Economic Perspectives*, 14(3):159–181, September 2000.

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

Gill, J., Tinckam, K., Fortin, M., Rose, C., Shick-Makaroff, K., Young, K., Lesage, J., Cole, E., Toews, M., Landsberg, D., and Gill, J. Reciprocity to increase participation of compatible living donor and recipient pairs in kidney paired donation. *Am J Transplant.*, 17(7), 2017.

Gouldner, A. The norm of reciprocity: A preliminary statement. *American Sociological Review*, (2):161–178, 1960.

Guibert, J. and Rayón, A. Uk's turing scheme. *International Higher Education*, (106):23–24, Apr. 2021.

Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., Venugopalan, S., Widner, K., Madams, T., Cuadros, J., Kim, R., Raman, R., Nelson, P. Q., Mega, J., and Webster, D. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *JAMA*, 2016.

Hara, S., Nitanda, A., and Maehara, T. Data cleansing for models trained with sgd. In *Advances in Neural Information Processing Systems*, pp. 4215–4224, 2019.

Harper, F. M. and Konstan, J. A. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5 (4), December 2015.

Hoaglin, D. C. and Welsch, R. E. The hat matrix in regression and anova. *The American Statistician*, 32(1):17–22, 1978.

Jia, R., Dao, D., Wang, B., Hubis, F. A., Hynes, N., Gürel, N. M., Li, B., Zhang, C., Song, D., and Spanos, C. J. Towards efficient data valuation based on the shapley value. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1167–1176, 2019.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Nitin Bhagoji, A., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., El Rouayheb, S., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konecný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Qi, H., Ramage, D., Raskar, R., Raykova, M., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. Advances and open problems in federated learning. *Found. Trends Mach. Learn.*, 14(1–2):1–210, jun 2021.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1885–1894, 2017a.

Koh, P. W. and Liang, P. Understanding black-box predictions via influence functions. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1885–1894. PMLR, 06–11 Aug 2017b.

Koren, Y. and Bell, R. *Advances in Collaborative Filtering*, pp. 77–118. Springer US, Boston, MA, 2015.

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

Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., and Galstyan, A. A survey on bias and fairness in machine learning. *ACM Comput. Surv.*, 54(6), jul 2021.

Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pp. III–1310–III–1318. JMLR.org, 2013.

Perez, M. V., Mahaffey, K. W., Hedlin, H., Rumsfeld, J. S., Garcia, A., Ferris, T., Balasubramanian, V., Russo, A. M., Rajmane, A., Cheung, L., Hung, G., Lee, J., Kowey, P., Talati, N., Nag, D., Gummidipundi, S. E., Beatty, A., Hills, M. T., Desai, S., Granger, C. B., Desai, M., and Turakhia, M. P. Large-scale assessment of a smartwatch to identify atrial fibrillation. *New England Journal of Medicine*, 381(20):1909–1917, 2019.

Pruthi, G., Liu, F., Kale, S., and Sundararajan, M. Estimating training data influence by tracing gradient descent. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 19920–19930. Curran Associates, Inc., 2020.

Rajkomar, A., Oren, E., Chen, K., Dai, A. M., Hajaj, N., Liu, P. J., Liu, X., Sun, M., Sundberg, P., Yee, H., Zhang, K., Duggan, G. E., Flores, G., Hardt, M., Irvine, J., Le, Q. V., Litsch, K., Marcus, J., Mossin, A., Tansuwan, J., Wang, D., Wexler, J., Wilson, J., Ludwig, D., Volchenboum, S. L., Chou, K., Pearson, M., Madabushi, S., Shah, N. H., Butte, A. J., Howell, M., Cui, C., Corrado, G., and Dean, J. Scalable and accurate deep learning for electronic health records. *CoRR*, abs/1801.07860, 2018.

Rashed, A., Grabocka, J., and Schmidt-Thieme, L. Attribute-aware non-linear co-embeddings of graph features. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, pp. 314–321, New York, NY, USA, 2019. Association for Computing Machinery.

Sundararajan, M. and Najmi, A. The many shapley values for model explanation. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9269–9278. PMLR, 13–18 Jul 2020.

Wang, D., Khosla, A., Gargeya, R., Irshad, H., and Beck, A. H. Deep learning for identifying metastatic breast cancer, 2016.

Yeh, C.-K., Kim, J., Yen, I. E.-H., and Ravikumar, P. K. Representer point selection for explaining deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 9291–9301, 2018.

Zhang, S., Yao, L., and Xu, X. Autosvd++: An efficient hybrid collaborative filtering model via contractive auto-encoders. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pp. 957–960, New York, NY, USA, 2017. Association for Computing Machinery.

Appendix

A. Reciprocity in Recommender Systems

We give additional details about reciprocity in the context of a standard recommendation model (Bobadilla et al., 2013). There is a population U of individuals (indexed by u) and a set I of items (pieces of content, indexed by i). When individuals interact with items, they *rate* the item implicitly or explicitly. This produces a score r_{ui}. The individuals and the items may have features associated with them. These could be ids, or descriptive features such as the genre of the movie, the location of the individual etc. The task is to predict the scores for unseen interactions given scores for past interactions.

We seek to measure the value exchange from the implicit *curation* that occurs when individuals consume recommendations. For a recommender system, the features (x from Section 2.1 are characteristics of individuals and items, and the label (y from Section 2.1) is a rating r_{ui} for the individual-item pair u, i associated with the example.

In such a model, every individual u is endowed with a d dimensional vector p_u and every item i is endowed with a d dimensional vector q_i; we refer to these vectors as embeddings. The prediction matrix is the product $\hat{R} = PQ^\top$, where p_u is the u-th row of matrix P and similarly for Q. In other words, the predicted rating for user-item pair (u, i) is given by the dot product $\hat{r}_{u,i} = p_u \cdot q_i$. See (Koren et al., 2009) for more details about matrix factorization.

The model is optimized for a regularized quadratic loss, i.e.,

$$\frac{1}{2} \sum_{(u,i)\in Z} (p_u \cdot q_i - r_{ui})^2 + \frac{\lambda}{2} \left(\sum_u \|p_u\|^2 + \sum_i \|q_i\|^2 \right),$$

where r_{ui} is the label of pair (u, i). The regularization term helps generalization.

For matrix factorization, we redistribute the regularization term as a sum over training examples, and define the loss as

$$\frac{1}{2} \sum_{(u,i)\in Z} \left((p_u \cdot q_i - r_{ui})^2 + \frac{\lambda}{|Z_u|} \|p_u\|^2 + \frac{\lambda}{|Z_i|} \|q_i\|^2 \right),$$

where $Z_u = \{i : (u, i) \in Z\}$ and $Z_i = \{u : (u, i) \in Z\}$.

Remark A.1 (`TracIn` for Matrix Factorization). Notice that the loss gradients and the `TracIn` influence have a simple structure. A visit to a training example (u, i) only updates the vectors p_u and q_i. Moreover, the change in the user embedding p_u only affects the predictions for user u; therefore updates to the user vectors do not play a role in the definitions of inflow and outflow. (Equations 1 and 2).

The update to the item embedding q_i only influences users who interact with the item i in the inference set. Thus, for this loss function, inflows and outflows *only flow through updates to item embeddings*.

Remark A.2 (Computing `TracIn` for Matrix factorization). Section 4 suggests that we can compute `TracIn` in $O(|Z|p + |Z'|p)$ where p is the total number of model parameters, in this case $p = d(|U| + |I|)$. But due to the structure of the problem, this computation can be done more efficiently for matrix factorization. Notice that the gradient of the loss w.r.t. a training example $z = (u, i)$ is $2d$-sparse, only the embeddings p_u, q_i have a non-zero gradient. Thus, computing $\nabla L_Z(w_t)$ in Equation (6), can be done in $O(|Z|d)$ instead of $O(|Z|p)$. Similarly, computing the sum of dot products in Equation (5) requires $O(|Z'|d)$ operations (since each $\nabla L_z(w_t)$ is $2d$-sparse). The total complexity is therefore $O(|Z|d + |Z'|d)$. This is equal to the complexity of running gradient descent, which means that computing Inflows and Outflows along the SGD trajectory does not significantly increase the computational cost of model training.

B. Proof of Theorem 3.2

First, we introduce some notation. Let $\mathcal{Z} = U \times X \times Y$, where U is the population of individuals, X is the feature set and Y is the label set. Let D be the joint distribution over $\mathcal{Z}$. For an individual $u \in U$, we write $\mathcal{Z}_u = \{u\} \times X \times Y$, so that a training example z belongs to individual u if $z \in \mathcal{Z}_u$.

Our goal is to show that for all u, $\mathbb{E}[I_u] = \mathbb{E}[O_u]$.

Given batches of training data $B_1, \ldots, B_T$ and an inference set Z' (note that both are random variables), we rewrite inflows and outflows in a form that is more amenable to taking expectations.

$$\begin{aligned}
I_u &= \sum_{t=1}^{T} \eta_t \sum_{z \in B_t : z \notin \mathcal{Z}_u} \sum_{z' \in Z'_u} \nabla L_z(w_t) \cdot \nabla L_{z'}(w_t) \\
&= \sum_{t=1}^{T} \eta_t \left(\sum_{z \in B_t} \nabla L_z(w_t) 1_{[z \notin \mathcal{Z}_u]} \right) \cdot \left(\sum_{z' \in Z'} \nabla L_{z'}(w_t) 1_{[z' \in \mathcal{Z}_u]} \right), \qquad (7)
\end{aligned}$$

where $1_{[z' \in \mathcal{Z}_u]}$ is the indicator of the event "z' belongs to user u". Similarly, we have for outflows

$$O_u = \sum_{t=1}^{T} \eta_t \left(\sum_{z \in B_t} \nabla L_z(w_t) 1_{[z \in \mathcal{Z}_u]} \right) \cdot \left(\sum_{z' \in Z'} \nabla L_{z'}(w_t) 1_{[z' \notin \mathcal{Z}_u]} \right). \qquad (8)$$

Let $(F_1, \dots, F_t)$ denote the filtration arising from the sequence of random variables $(B_1, \dots, B_t)$. Taking the expectation of inflow in Equation (7), and using the tower property of conditional expectations, we have

$$\mathbb{E}[I_u] = \mathbb{E}\left[\sum_{t=1}^{T} \eta_t \, \mathbb{E}\left[\sum_{z \in B_t} \nabla L_z(w_t) 1_{[z \notin \mathcal{Z}_u]} \cdot \sum_{z' \in Z'} \nabla L_{z'}(w_t) 1_{[z' \in \mathcal{Z}_u]} \middle| F_{t-1} \right] \right].$$

Now, notice that the batches $(B_1, \dots, B_{t-1})$ completely determine the model parameters w_t (since $w_t = w_0 - \sum_{\tau=0}^{t-1} \eta_\tau \sum_{z \in B_\tau} \nabla L_z(w_\tau)$), and by assumption, the next batch of training examples B_t is independent of previous batches, and so is the inference set Z'. So conditioned on F_{t-1}, the two random variables $\sum_{z \in B_t} \nabla L_z(w_t) 1_{[z \notin \mathcal{Z}_u]}$ and $\sum_{z' \in Z'} \nabla L_{z'}(w_t) 1_{[z' \in \mathcal{Z}_u]}$ are independent. Let us denote by

$$\begin{aligned}
g_{t-1}^{-u} &= \mathbb{E}_{z \sim D}[\nabla L_z(w_t) 1_{[z \notin \mathcal{Z}_u]} | F_{t-1}], \\
g_{t-1}^{u} &= \mathbb{E}_{z' \sim D}[\nabla L_{z'}(w_t) 1_{[z' \in \mathcal{Z}_u]} | F_{t-1}].
\end{aligned}$$

Then, by the aforementioned independence, linearity of expectations, and the assumption that elements of Z' and Z (and hence B_t) follow the same distribution D, we have

$$\mathbb{E}[I_u] = \mathbb{E}\left[\sum_{t=1}^{T} \eta_t (|B_t| g_{t-1}^{-u}) \cdot (|Z'| g_{t-1}^{u}) \right]. \qquad (9)$$

We make a similar calculation for outflows (the only difference is in the indicators): taking expectations in Equation (8),

$$\begin{aligned}
\mathbb{E}[O_u] &= \mathbb{E}\left[\sum_{t=1}^{T} \eta_t \, \mathbb{E}\left[\sum_{z \in B_t} \nabla L_z(w_t) 1_{[z \in \mathcal{Z}_u]} \cdot \sum_{z' \in Z'} \nabla L_{z'}(w_t) 1_{[z' \notin \mathcal{Z}_u]} \middle| F_{t-1} \right] \right], \\
&= \mathbb{E}\left[\sum_{t=1}^{T} \eta_t (|B_t| g_{t-1}^{u}) \cdot (|Z'| g_{t-1}^{-u}) \right], \qquad (10)
\end{aligned}$$

where we used independence (conditional on F_{t-1}) of the random variables $\sum_{z \in B_t} \nabla L_z(w_t) 1_{[z \in \mathcal{Z}_u]}$ and $\sum_{z' \in Z'} \nabla L_{z'}(w_t) 1_{[z' \notin \mathcal{Z}_u]}$. The two quantities (9) and (10) are equal. This concludes the proof.

C. Additional Experimental Results

C.1. Sanity Checks on MovieLens

In order to ensure that inflows and outflows are meaningful quantities, we measure the noise in the inflows and outflows due to randomness in the training process.

We plot the distributions across five arbitrarily chosen individuals. See Figure 4. To aid interpretation, we normalize the inflows and outflows by the total inflow (which approximates the total change in loss). Therefore, one can read the numbers as the fraction of overall inflow, (or approximately the fraction of total loss reduction). We observe that the inflows and outflows are consistent across runs, however, there is some variation, stemming from the random initialization.

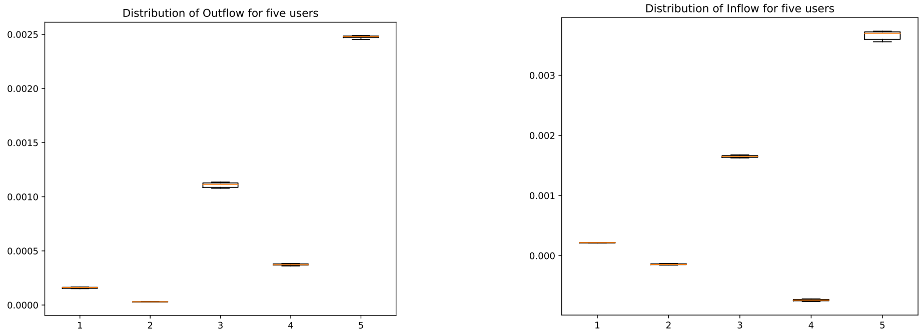


Figure 4. Distribution of inflows and outflows across 10 runs for 5 individuals. The box edges show the upper and lower quartiles, the whiskers show the fifth and ninety-fifth percentile.

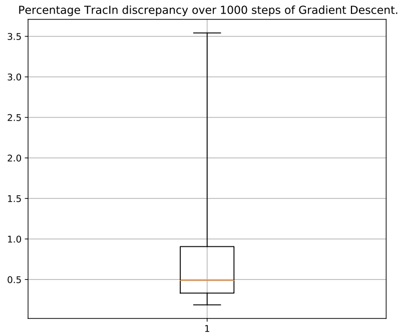


Figure 5. `TracIn` approximation discrepancy across 1000 steps of Gradient Descent.

Next, we measure the error induced by the first-order approximation of `TracIn` (recall Remark 2.5). Figure 5 shows the relative error for one run. The numerator is the sum of the gradient dot products (in Equation 4) across examples for one step of gradient descent minus the total change in loss across Z'. The denominator is the total change in loss across Z'. The percentage relative discrepancy remains relatively small; its 80th percentile is 1.1%. This can be further reduced by using a smaller learning rate.

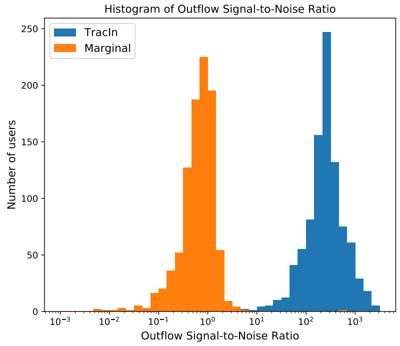


Figure 6. Histogram of user outflow signal-to-noise ratios, computed across ten splits.

C.2. Marginal Influence on the MovieLens Data Set

In the MovieLens experiment, we found that `Marginal`-outflows and inflows are noisy, and extremely susceptible to random initialization (recall Remark 2.4). This can be quantified by measuring the Signal-to-Noise Ratio (SNR), defined as the mean divided by the standard deviation. In an attempt to improve the SNR, we used a slightly different definition of marginal flows: instead of measuring the effect of deleting a single example, as in Equation (3), then summing over $z \in Z_u$, we measure the effect of removing all of user u's examples simultaneously. Despite this change, the `Marginal` SNR remains low, see Figure 6. For `Marginal`, over 99% of individuals have a SNR less than one. For `TracIn`, 90% of individuals have a SNR greater than 76, making `TracIn` measurements much more meaningful.