Self-Similar Epochs: Value in Arrangement

Eliav Buchnik [*1,2] **Edith Cohen** [*2,1] **Avinatan Hassidim** [2] **Yossi Matias** [2]

Abstract

Optimization of machine learning models is commonly performed through stochastic gradient updates on randomly ordered training examples. This practice means that each fraction of an epoch comprises an independent random sample of the training data that may not preserve informative structure present in the full data. We hypothesize that the training can be more effective with *self-similar* arrangements that potentially allow each epoch to provide benefits of multiple ones. We study this for "matrix factorization" – the common task of learning metric embeddings of entities such as queries, videos, or words from example pairwise associations. We construct arrangements that preserve the weighted Jaccard similarities of rows and columns and experimentally observe training acceleration of 3%-37% on synthetic and recommendation datasets. Principled arrangements of training examples emerge as a novel and potentially powerful enhancement to SGD that merits further exploration.

1. Introduction

Large scale machine learning models are commonly trained on data of the form of associations between entities. The goals are to obtain a model that generalizes (supports inference of associations not present in the input data) or obtain metric representations of entities that capture their associations and can be used as in downstream tasks. Prevalent examples are images and their labels (Deng et al., 2009), similar image pairs (Schroff et al., 2015), text documents and occurring terms (Berry et al., 1995; Dumais, 1995; Deerwester et al., 1990), users and watched or rated videos (Koren et al., 2009), pairs of co-occurring words (Mikolov et al., 2013), and pairs of nodes in a graph that co-occur in short random walks (Perozzi et al., 2014). This setting is fairly broad: entities can be of one or multiple types and example associations used for training can be raw or preprocessed by reweighing raw frequencies (Salton & Buckley, 1988; Deerwester et al., 1990; Mikolov et al., 2013; Pennington et al., 2014) or adding negative examples when the raw data includes only positive ones (Koren et al., 2009; Mikolov et al., 2013).

The optimization objective of the model parameters has the general form of a sum over example associations. In modern applications the number of terms can be huge and the de facto method is stochastic gradient descent (SGD) (Robbins & Siegmund, 1971; Koren, 2008; Salakhutdinov et al., 2007; Gemulla et al., 2011; Mikolov et al., 2013). With SGD, gradient updates computed over stochastically-selected minibatches of training examples are performed over multiple epochs. The extensive practice and theory of SGD optimization introduced numerous tunable hyper-parameters and extensions aimed to improve quality and efficiency. These include tuning the learning rate also per-parameter (Duchi et al., 2011) and altering the distribution of training examples by gradient magnitudes (Alain et al., 2015; Zhao & Zhang, 2015), cluster structure (Fu & Zhang, 2017), and diversity criteria (Zhang et al., 2017). Another popular method, Curriculum Learning (Bengio et al., 2009), alters the distribution of examples (from easy to hard) in the course of training. In this work we motivate and explore the potential benefits of tuning the *arrangement* of training examples – a novel optimization method that is combinable with those mentioned above.

The baseline practice, which we refer to as *independent* arrangements, forms the training order by drawing examples (or minibatches of examples) independently at random according to prescribed probabilities (that may correspond to the frequency of the association in the training set). This i.i.d practice is supported by optimization theory as it bounds the variance of stochastic gradient updates. Independent arrangements, however, have a potential drawback: Informative structure that is present in the full data is not recoverable from the independent random samples that comprise fractional epochs. We demonstrate this point for a common "matrix factorization" task where the data is pairwise associations such as users watches of videos for recommendation tasks (Koren et al., 2009) and term co-occurrences for word embeddings (Mikolov et al., 2013). The rows and

[*]Equal contribution [1]Tel Aviv University, Israel [2]Google Research. Correspondence to: Eliav Buchnik <eliavbuh@gmail.com>, Edith Cohen <edith@cohenwang.com>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

columns represent entities and entries are example associations (viewers $\times$ videos or word $\times$ word). In such data

Independent arrangement Coordinated arrangement

Figure 1. Two rows (A and B), where entries in red are positive and equal. The Jaccard similarity is $J(A, B) = 1/2$ (5 common positive columns out of 10 that are positive for at least one). A fraction of an epoch includes a random sample from each row. With independent arrangements the 4 samples from each row are unlikely to align on the common columns resulting in empirical Jaccard similarity of 0. With our coordinated arrangement the samples align and the empirical Jaccard similarity is $1/2$.

(see Figure 1), the similarity of two rows (or columns) is indicative of the similarity of the corresponding entities that our model is out to capture. For example, two videos with overlapping sets of viewers are likely to be similar. While the target similarity we seek is typically more complex and in particular reflects higher order relations (sets of similar but not overlapping viewers), this "first order" similarity is nonetheless indicative. In a random sample of matrix entries, however, two similar rows will have dissimilar samples: The expected empirical weighted Jaccard similarity on the sample is much lower than the respective similarity in the data, and this happens even at the extreme where the sample is a large fraction of the full data (say half an epoch) and we are considering two identical rows (!). For our training this means that fractions of epochs rapidly lose this important information that is present in the full dataset. We hypothesize that this may impact the effectiveness of training: An arrangement that is more "self-similar" in the sense that information is preserved to a higher extent in fractional epochs may allow a single epoch to provide benefits of multiple ones and for the training to converge faster.

We approach this by designing *coordinated* arrangements that preserve in expectation in fractional epochs the weighted Jaccard similarities of rows and columns. Our design is inspired by the theory of coordinated weighted sampling (Kish & Scott, 1971; Brewer et al., 1972; Cohen et al., 2009; Cohen, 2014) which are related to MinHash sketches (Cohen, 1997; Broder, 2000). In coordinated sampling the goal is to select samples of entries of vectors that can provide more accurate estimates of relations between the vectors than independent samples. In our application here we will construct arrangements of examples where subsequences look like coordinated samples.

We specify our coordinated arrangements by a distribution on randomized subsets of example associations which we refer to as *microbatches*. Our training sequence consists of independent microbatches and thus retains the traditional advantages of i.i.d training at the coarser microbatch level. Note that our microbatches are designed so that the probability that each example is placed in a microbatch is equal to its prespecified baseline marginal probability. Therefore, the only difference between coordinated and independent arrangements is in the ordering.

In some applications or training regimes smaller microbatches, which allow for more independence, can be more effective. Our coordinated microbatches are optimized in size to preserve expected similarities. Microbatch sizes can be naively decreased by random partitions – but this break down the similarity approximation and more so for similar pairs, which are exactly the ones for which the benefits of preserving similarities are larger. We show how Locality Sensitive Hashing (LSH) maps can be used to decrease microbatch sizes in a targeted way that compromises more the less similar pairs. We explore LSH maps that leverage coarse available proxies of entity similarity: The weighted Jaccard similarity of the row and column vectors or angular similarity of an embedding obtained by a weaker model.

We design efficient generators of coordinated and LSH-refined microbatches and study the effectiveness of different arrangements through experiments on synthetic stochastic block matrices and on recommendation data sets. We use the popular Skip Gram with Negative Sampling (SGNS) loss objective (Mikolov et al., 2013). We observe consistent training gain of 12-37% on blocks and of 3%-12% on our real data sets when using coordinated arrangements.

The paper is organized as follows. Section 2 presents necessary background on the loss objective we use in our experiments and working with minibatches with one-sided gradient updates and selection of negative examples. In Section 3 we present our coordinated microbatches and in Section 4 we establish their properties. Our LSH refinements are presented in Section 5 and our experimental results are reported in Sections 6 and 7. We conclude in Section 8.

2. Preliminaries

Our data has the form of associations between a *focus* entity from a set F and a *context* entity from a set C. The focus and context entities can be of different types (users and videos) or two roles of the same type or even of the same set (as in word embeddings). We use κ_{ij} as the association strength between focus i and context j. In practice, the association strength can be derived from frequencies in the raw data or from an associated value (for example, numeric rating or watch time).

An embedding is a set of vectors $\boldsymbol{f}_i, \boldsymbol{c}_j \in \Re^d$ that is trained to minimize a loss objective that encourages $\boldsymbol{f}_i$ and $\boldsymbol{c}_j$ to be "closer" when κ_{ij} is larger. Examples of positive associations (i, j) are drawn with probability proportional to κ_{ij}. Random associations are then used as negative examples

(Hu et al., 2008) that provide an "antigravity" effect that prevents all embeddings from collapsing into the same vector. The weight

$$n_{ij} := \lambda \|\kappa_{\cdot j}\|_1 \|\kappa_{i\cdot}\|_1 / \|\kappa\|_1 \qquad (1)$$

of a negative example (i, j) is proportional to the product of its column sum $\|\kappa_{\cdot j}\|_1$ by its row sum $\|\kappa_{i\cdot}\|_1$. The hyperparameter λ specifies a ratio of negative to positive examples.

Our design applies to objectives of the general form

$$L := \sum_{ij} \kappa_{ij} L_+(\boldsymbol{f}_i, \boldsymbol{c}_j) + \sum_{ij} n_{ij} L_-(\boldsymbol{f}_i, \boldsymbol{c}_j) \qquad (2)$$

and can also accomodate hidden parameters as in (Bromley et al., 1994; Chopra et al., 2005). For concreteness, we focus here on Skip Gram with Negative Sampling (SGNS) (Mikolov et al., 2013). The SGNS objective is designed to maximize the log likelihood of these examples. The probability of positive and negative examples are respectively modeled using

$$\begin{aligned} p_{ij} = \quad & \sigma(\boldsymbol{f}_i \cdot \boldsymbol{c}_j) = \frac{1}{1 + \exp(-\boldsymbol{f}_i \cdot \boldsymbol{c}_j)} \\ 1 - p_{ij} = \quad & \sigma(-\boldsymbol{f}_i \cdot \boldsymbol{c}_j) = \frac{1}{1 + \exp(\boldsymbol{f}_i \cdot \boldsymbol{c}_j)} \,. \end{aligned}$$

The likelihood function, which we seek to maximize, can then be expressed as $\Pi_{ij}\, p_{ij}^{\kappa_{ij}}\, \Pi_{ij}(1 - p_{ij})^{n_{ij}}$. We equivalently can minimize the negated log likelihood that turns the objective into a sum of the form (2):

$$L := -\sum_{ij} \kappa_{ij} \log p_{ij} - \sum_{ij} n_{ij} \log(1 - p_{ij}) \,.$$

(using $L_+(\boldsymbol{f}_i, \boldsymbol{c}_j) := -\log \sigma(\boldsymbol{f}_i \cdot \boldsymbol{c}_j)$ and $L_-(\boldsymbol{f}_i, \boldsymbol{c}_j) := -\log \sigma(-\boldsymbol{f}_i \cdot \boldsymbol{c}_j)$.)

The optimization is performed by random initialization of the embedding vectors followed by stochastic gradient updates. The stochastic gradients are computed for minibatches of examples that include b positive examples, where (i, j) appears with frequency $\kappa_{ij}/\|\kappa\|_1$ and a set of $b\lambda$ negative examples.

2.1. One-sided updates

We work with *one-sided* updates, where each minibatch updates only its focus or only its context embedding vectors, and accordingly say that minibatches are *designated* for focus or context updates. One-sided updates are used with alternating minimization (Csiszar & Tusnády, 1984) and decomposition-coordination approaches (Cohen, 1980). For our purposes, one-sided updates facilitate our coordinated arrangements (intuitively, because we need to separately preserve column and row similarities) and also allow for precise minibatch-level matching of each positive update of a parameter with a corresponding set of negative updates as a means to control variance.

Our minibatches are constructed from a set P of b positive examples and matched negatives. Our marginal probabilities of positive and negative examples (see Eq. 1) are equivalent to pairing each positive example (i, j) (with marginal probability $\kappa_{ij}/\|\kappa\|_1$) with (i) λ negative examples of the form (i, j') where j' is a random context entities (selected proportionally to the column sum $\|\kappa_{\cdot j'}\|/\|\kappa\|_1$ and (ii) λ negative examples of the form (i', j) where i' are random focus entities i' (selected proportionally to their row sums $\|\kappa_{i'\cdot}\|_1/\|\kappa\|_1$). With one-sided updates, we pair each positive example $(i, j) \in P$ with λ negative examples selected according to the respective designation. To form a focus-updating minibatch, we generate a random set of λ context vectors C'. For each positive example $(i, j) \in P$ we generate λ negative examples (i, j') for $j' \in C'$. The focus embedding $\boldsymbol{f}_i$ is updated to be closer to $\boldsymbol{c}_j$ but at the same time repealed (in expectation) from C' context vectors. With learning rate η, the combined update to $\boldsymbol{f}_i$ due to positive example (i, j) and matched negatives is

$$\Delta \boldsymbol{f}_i = -\eta \nabla_{\boldsymbol{f}_i} \left(L_+(\boldsymbol{f}_i, \boldsymbol{c}_j) + \sum_{j' \in C'} L_-(\boldsymbol{f}_i, \boldsymbol{c}_{j'}) \right) \,.$$

Symmetrically, to form a context-updating minibatch we draw a random set of focus vectors F' and generate respective negative examples. Each positive example $(i, j) \in P$ yields an update of context vector $\boldsymbol{c}_j$ by $\Delta \boldsymbol{c}_j = -\eta \nabla_{\boldsymbol{c}_j} \left(L_+(\boldsymbol{f}_i, \boldsymbol{c}_j) + \sum_{i' \in F'} L_-(\boldsymbol{f}_{i'}, \boldsymbol{c}_j) \right)$. All updates are combined and applied at the end of the minibatch.

3. Arrangement Schemes

Arrangement schemes determine how examples are organized. At the core of each scheme is a distribution $\mathcal{B}$ over subsets of positive examples which we call *microbatches*. Our microbatch distributions have the property that the marginal probability of each example (i, j) is always equal to $\kappa_{ij}/\|\kappa\|_1$ but subset probabilities vary across schemes. Moreover, within a scheme we may have different distributions $\mathcal{B}_f$ for focus and $\mathcal{B}_c$ for context designations.

Minibatches formation for focus updates is specified in Algorithm 1 (the construction for context updates is symmetric). The input is a microbatch distribution $\mathcal{B}_f$, minibatch size parameter b, and a parameter λ that determines the ratio of negative to positive training examples. We draw independent microbatches until we have a total of b or more positive examples and then select negative examples as described above. When training, we alternate between focus and context updating minibatches to maintain balance between the total number of examples processed with each designation.

Algorithm 1: Minibatch construction (Focus updates)

```
Input: B_f, b, λ // Microbatch distribution, size, negative sampling
P, N ← ∅
repeat X ∼ B_f; P ← P ∪ X
 until |P| ≥ b
C′ ← λ contexts selected iid by column weights
foreach example pair (i, j) ∈ P do
    foreach j′ ∈ C′ do
        N ← N ∪ {(i, j′)}
return P ∪ N
```

The baseline independent arrangement method (IND) can be placed in this framework using microbatches that consist of a single positive example (i, j) selected with probability $\kappa_{ij}/\|\kappa\|_1$ (see Algorithm 2). Our coordinated microbatches (COO) have different distributions for focus and context updates. Algorithm 3 generates focus microbatches (the generator for context designation is symmetric). These microbatches have the form of a set of positive examples with a shared context. In the instructive special case of κ with all-equal positive entries focus microbatches include all positive entries in some column and context microbatches include all positive entries in a raw.

We preprocess κ so that we can efficiently draw j with probability $\|\kappa_{\cdot j}\|_\infty / \sum_h \|\kappa_{\cdot h}\|_\infty$ and construct an index that for context j and value T efficiently returns all entries i with $\kappa_{ij} \geq T$. The preprocessing is linear in the sparsity of κ and with it the microbatch generator amounts to drawing a context j (an$O(1)$ operation), $u \sim U[0, 1]$ and then query the index with j and $T = u\|\kappa_{\cdot j}\|_\infty$. The preprocessing cost for microbatch generation is often dominated by the preprocessing done to generate κ from raw data.

Algorithm 2: IND microbatches

```
Input: κ
Choose (i, j) with probability κ_ij/‖κ‖_1;
return {(i, j)}
```

Algorithm 3: COO microbatches (Focus updates)

```
Input: κ
// Preprocessing:
foreach context j do
    M_j ← max_i κ_ij // Maximum entry for context j
    Index column j so that we can return for each t ∈ (0, 1],
      P(j, t) := {i | κ_ij ≥ t M_j}.
// Microbatch draw:
Choose a context j with probability M_j / Σ_h M_h
Draw u ∼ U[0, 1]
return {(i, j) | i ∈ P(j, u)}
```

4. Properties of COO Arrangements

We establish that COO arrangements produce the same marginal distribution on training examples as the baseline IND arrangements. We then highlight two properties of coordinated arrangements that are beneficial to accelerating convergence: A micro-level property that makes gradient updates more effective by moving embedding vectors of similar entities closer and a macro-level property of preserving expected similarity in fractions of epochs.

Marginal distribution We show that the frequency of each example (i, j) to occur in a COO microbatch is the same in both designations and $\propto \kappa_{ij}$.

Lemma 4.1. *The inclusion probability of a positive example* (i, j) *in a coordinated microbatch with focus designation (Algorithm 3) is* $\kappa_{ij}/\sum_h \|\kappa_{\cdot h}\|_\infty$*, where the notation* $\|\kappa_{\cdot h}\|_\infty$ *is the maximum entry in column* h*. Respectively, the inclusion probability of* (i, j) *in a microbatch with context designation is* $\kappa_{ij}/\sum_h \|\kappa_{h\cdot}\|_\infty$*, where* $\|\kappa_{h\cdot}\|_\infty$ *is the maximum entry at row* h*.*

Proof. Consider focus updates (apply a symmetric argument for context updates). The example (i, j) is selected when first context j is selected, which happens with probability $\|\kappa_{\cdot j}\|_\infty/\sum_h \|\kappa_{\cdot h}\|_\infty$ and then we have $u \leq \kappa_{ij}/\|\kappa_{\cdot j}\|_\infty$ for independent $u \sim U[0, 1]$, which happens with probability $\kappa_{ij}/\|\kappa_{\cdot j}\|_\infty$. Combining, the probability that (i, j) is selected is the product of the probabilities of these two events which is $\kappa_{ij}/\sum_h \|\kappa_{\cdot h}\|_\infty$. □

Our arrangements consist of both focus and context microbatches that balance the total number of examples in each designation. Therefore, each example appears in the same frequency with each designation.

Alignment of corresponding examples We establish that our COO microbatches maximize the co-placement probability of *corresponding* pairs of examples (examples with shared context or focus) and discuss why this is helpful in accelerating training.

Lemma 4.2. *If a focus-designation* COO *microbatch includes an example* (i, j) *and* $\kappa_{ij} \leq \kappa_{i'j}$ *then it also includes the example* (i', j)*. Symmetrically with context-designation,* (i, j) *being included and* $\kappa_{ij} \leq \kappa_{ij'}$ *implies that* (i, j') *is also included.*

We show that aligned updates on corresponding examples are helpful in bringing the embedding vectors closer. A pair of entities with higher Jaccard similarity has more corresponding examples and benefit more from alignment. Interestingly, the benefit is there even when embedding vectors are random (as is the case early in training). In particular, the SGNS loss term for a positive example is $L_+(\boldsymbol{f}, \boldsymbol{c}) = -\log \sigma(\boldsymbol{f}, \boldsymbol{c}) = -\log\left(\frac{1}{1+\exp(-\boldsymbol{f}\cdot\boldsymbol{c})}\right)$. The gradient with respect to $\boldsymbol{f}$ is $\nabla_{\boldsymbol{f}}(L_+(\boldsymbol{f}, \boldsymbol{c})) = -\boldsymbol{c}\frac{1}{1+\exp(\boldsymbol{f}\cdot\boldsymbol{c})}$ and the respective update of $\boldsymbol{f}' \leftarrow \boldsymbol{f} + \eta\frac{1}{1+\exp(\boldsymbol{f}\cdot\boldsymbol{c})}\boldsymbol{c}$ clearly increases $\cos_{\text{sim}}(\boldsymbol{f}, \boldsymbol{c})$.

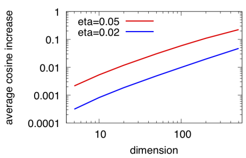

Figure 2. Expected increase in $\cos_{\text{sim}}(\boldsymbol{f}_1, \boldsymbol{f}_2)$ as a function of dimension for $\boldsymbol{f}_i \sim \mathcal{N}^d$ after gradient update to same random context $\boldsymbol{c} \sim \mathcal{N}^d$

Consider two focus entities $1, 2$ and corresponding examples $(1, j)$ and $(2, j)$. When the two examples are in the same focus-updating minibatch (where $\boldsymbol{c}_j$ is fixed) both $\cos_{\text{sim}}(\boldsymbol{f}_1, \boldsymbol{c})$ and $\cos_{\text{sim}}(\boldsymbol{f}_2, \boldsymbol{c})$ increase but a desirable side effect is that in expectation $\cos_{\text{sim}}(\boldsymbol{f}_1, \boldsymbol{f}_2)$ increases as well. The updates are aligned also with full gradients but not with IND arrangements that on average place corresponding examples half an epoch apart. Figure 2 shows the expected increase in cosine similarity $\mathbb{E}\left[\cos_{\text{sim}}(\boldsymbol{f}'_1, \boldsymbol{f}'_2) - \cos_{\text{sim}}(\boldsymbol{f}_1, \boldsymbol{f}_2)\right]$ as a function of the dimension for example learning rates $\eta = 0.02, 0.05$ when the vectors $\boldsymbol{f}_1$, $\boldsymbol{f}_2$, and $\boldsymbol{c}$ are independently drawn from a product distribution $\mathcal{N}(0, 1)^d$ of independent Gaussians.

Preservation of Jaccard similarities We establish that COO arrangements preserve in expectation Jaccard similarities of pairs of rows and columns. The weighted Jaccard similarity of two vectors $\boldsymbol{v}$ and $\boldsymbol{u}$ is defined as

$$J(\boldsymbol{v}, \boldsymbol{u}) = \frac{\sum_i \min\{v_i, u_i\}}{\sum_i \max\{v_i, u_i\}} . \tag{3}$$

Lemma 4.3. *Consider a set of focus updating microbatches and let X_{ij} be the random variable that is the multiplicity of example (i, j). Then for any two rows i, i', the expectation of the empirical weighted Jaccard similarity on X (when defined) is equal to the weighted Jaccard similarity on κ:*

$$\mathbb{E}\left[J(X_{i\cdot}, X_{i'\cdot}) \mid \sum_j \max\{X_{i',j}, X_{i,j}\} > 0\right] = J(\kappa_{i,\cdot}, \kappa_{i',\cdot})$$

A symmetric claim holds for context updating microbatches.

Proof. We consider a single microbatch and its contributions to the numerator and denominator of the empirical similarity $J(X_{i\cdot}, X_{i'\cdot})$. From Lemma 4.2, the possible contributions are $(0, 0)$, $(0, 1)$ or $(1, 1)$. Therefore, $J(X_{i\cdot}, X_{i'\cdot})$ (if defined) is simply the average of the contributions to the numerator over microbatches that contributed to the denominator. The expectation of $J(X_{i\cdot}, X_{i'\cdot})$ (when defined) is therefore equal to the probability of a contribution to the numerator in a single microbatch given that there was a contribution to the denominator. If the shared context in the microbatch is j, the probability of contribution to the denominator is $\max\{\kappa_{i',j}, \kappa_{i,j}\}/\|\kappa_{\cdot j}\|_\infty$ and to the numerator is $\min\{\kappa_{i',j}, \kappa_{i,j}\}/\|\kappa_{\cdot j}\|_\infty$. The probability over the random draw of context j of a contribution to the denominator and numerator respectively is $\sum_j \max\{\kappa_{i',j}, \kappa_{i,j}\}/\sum_j \|\kappa_{\cdot j}\|_\infty$ and $\sum_j \min\{\kappa_{i',j}, \kappa_{i,j}\}/\sum_j \|\kappa_{\cdot j}\|_\infty$. Since a contribution to the numerator is made only if there was one to the denominator, the expectation we seek is the ratio $J(\kappa_{i,\cdot}, \kappa_{i',\cdot})$. □

5. Refinement using LSH Maps

We provide methods to partition our COO microbatches so that they are smaller and of higher quality in the sense that a larger fraction of corresponding example pairs are between entities with higher similarity. To do this we use locality sensitive hashing (LSH) to compute randomized maps of entities to keys. Each map is represented by a vector $\mathbf{s}$ of keys for entities such that similar entities are more likely to obtain the same key. We use these maps to refine our basic microbatches by partitioning them according to keys.

Ideally, our LSH modules would correspond to the target similarity, but this creates a chicken-and-egg problem. Instead, we can use LSH modules that are available at the start of training and provide some proxy of the target similarity. For example, a partially trained or a weaker and cheaper to train model. We consider two concrete LSH modules based on *Jaccard* and on *Angular* LSH. The modules generate maps for either focus or context entities which are applied according to the microbatch designation. We will specify the map generation for focus entities, as maps for context entities can be symmetrically obtained by reversing roles.

Our Jaccard LSH module is outlined in Algorithm 4. The probability that two focus entities i and i' are mapped to the same key (that is, $s_i = s_{i'}$) is equal to the weighted Jaccard similarity of their association vectors $\kappa_{i\cdot}$ and $\kappa_{i'\cdot}$. (For context updates the map is according to the vectors $\kappa_{\cdot j}$):

Lemma 5.1. *(Cohen et al., 2009)*

$$\Pr[s_i = s_{i'}] = J(\kappa_{ij}, \kappa_{i'j})$$

Our angular LSH module is outlined in Algorithm 5. Here we input an explicit "coarse" embedding $\tilde{\boldsymbol{f}}_i, \tilde{\boldsymbol{c}}_j$ that we expect to be lower quality proxy of our target one. Each LSH map is obtained by drawing a random vector and then mapping each entity i to the sign of a projection of $\tilde{\boldsymbol{f}}_i$ on the random vector. The probability that two focus entities have the same key depends on the angle between their coarse embedding vectors:

Lemma 5.2. *(Goemans & Williamson, 1995)*

$$\Pr[s_i = s_{i'}] = 1 - \frac{1}{\pi}\cos^{-1}\cos_{sim}(\tilde{\boldsymbol{f}}_i, \tilde{\boldsymbol{f}}_{i'}) ,$$

where $\cos_{sim}(\boldsymbol{v}, \boldsymbol{u}) := \frac{\boldsymbol{v}\cdot\boldsymbol{u}}{\|\boldsymbol{v}\|_2\|\boldsymbol{u}\|_2}$ *is the cosine of the angle between the two vectors.*

Multiple LSH maps can be applied to decrease microbatch sizes and increase the similarity level of entities placed in the same microbatch: With r independent maps the probability that two entities are microbatched together is $\Pr[s_i = s_{i'}]^r$ – thus the probability decreases faster when similarity is lower. The number of LSH maps we apply can be set statically or adaptively to obtain microbatches that are at most a certain size (usually the minibatch size). For efficiency, we precompute a small number of LSH maps in the preprocessing step and randomly draw from that set. The computation of each map is linear in the sparsity of κ.

Algorithm 4: Jaccard LSH map: Focus

```
foreach context j do // i.i.d Exp distributed
    Draw u_j ~ Exp[1]
foreach focus i do // assign LSH bucket key
    s_i <- arg min_j u_j/κ_ij
return s
```

Algorithm 5: Angular LSH map: Focus

```
Input: {f̃_i} // coarse d dimensional embedding
Draw r ~ S_d // Random vector from the unit sphere
foreach focus i do // assign LSH bucket key
    s_i <- sign(r · f̃_i)
return s
```

6. Arrangement Methods Experiments

We trained embeddings with different arrangement methods: The baseline independent arrangements (IND) as in Algorithm 2, coordinated arrangements (COO) as in Algorithm 3, and some (COO+LSH) arrangements with Jaccard LSH. We also experimented with tunable arrangements (MIX) that start with COO (which reaps much of its benefit earlier in training) and switch to COO+LSH or to IND. Finally, as another baseline we also trained using the more standard IND with two-sided updates. The results were similar or slightly inferior to one-sided IND.

6.1. Stochastic blocks data

We generated data sets using the stochastic blocks model (Condon & Karp, 2001). This synthetic data allowed us to explore the effectiveness of different arrangement methods as we vary the number and size of blocks. The simplicity and symmetry of this data (parameters, entities, and associations) allowed us to compare different arrangement methods while factoring out optimizations and methods geared for asymmetric data such as per-parameter learning rates or altering the distribution of examples. The blocks data binary similarity allowed us to explore the limits of LSH refinements by refining COO microbatches according to ground truth similarity (partitioning COO microbatches by block membership) (COO+OPTLSH).

The parameters for the generative model are the dimensions $n \times n$ of the matrix, the number of (equal size) blocks B, the number of interactions r, and the in-block probability p. The rows and columns are partitioned to consecutive groups of n/B, where the ith part of rows and ith part of columns are considered to belong to the same block. We generate the matrix by initializing the associations to be $\kappa_{ij} = 0$. We then draw r interactions independently as follows. We select a row index $i \in [n]$ uniformly at random. With probability p, we select (uniformly at random) a column $j \in [m]$ that is in the same block as i and otherwise (with probability $1-p$) we select a column $j \in [n]$ that is outside the block of i. We then increment κ_{ij}. The final κ_{ij} is the number of times (i, j) was drawn. In our experiments we set $n = 10^4$, $r = 10^7$,$p = 0.7$ and $B \in \{10, 20, 50, 100\}$.

6.2. Implementation and methodology

We implemented our methods in Python using the TensorFlow library (Abadi & et al., 2015). We used the word embedding implementation of (Mikolov et al., 2013; word2vec.py) except that we used our methods to specify minibatches. The implementation included a default bias parameter associated with context embeddings and we trained embeddings with and without the bias parameter. The relative performance of arrangement methods was the same but the overall performance was significantly better when the bias parameter was used. We therefore report results with bias parameters. We used a fixed learning rate to facilitate a more accurate comparison of methods and trained with $\eta = 0.005$ to $\eta = 0.15$. We observed similar relative performance and report results with $\eta = 0.02$. We worked with minibatch size parameter values $b \in \{4, 64, 256\}$ (recall that b is the number of positive examples and $\lambda = 10$ negative examples are matched with each positive example), and embeddings dimension $d \in \{5, 10, 25, 50, 100\}$.

6.3. Quality measures

We use two measures of the quality of an embedding with respect to the blocks ground truth. The first is the *cosine gap* which measures average quality and is defined as the difference in the average cosine similarity between positive examples and negative examples. We generate a sampled

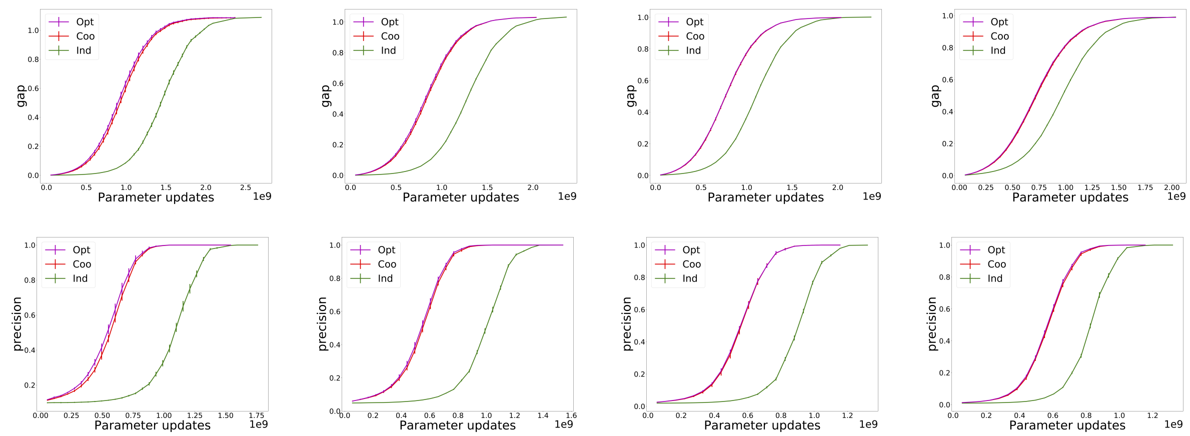

Figure 3. Cosine gap (top) and Precision at $k = 10$ (bottom) with COO and IND arrangements and COO+OPTLSH refinements in the course of training ($d = 50$, $b = 64$) for stochastic blocks with $B \in \{10, 20, 50, 100\}$ (left to right).

set T_+ of same-block pairs (i, j) as positive test examples and a sampled set $T\_$ of pairs that are not in the same block as negative test examples and compute

$$\frac{1}{|T_+|} \sum_{(i,j)\in T_+} \cos_{\text{sim}}(\boldsymbol{f}_i, \boldsymbol{c}_j) - \frac{1}{|T_-|} \sum_{(i,j)\in T_-} \cos_{\text{sim}}(\boldsymbol{f}_i, \boldsymbol{c}_j) \ . \tag{4}$$

We expect a good embedding to have high cosine similarity for same-block pairs and low cosine similarity for out of block pairs. The second measure we use, *precision at* k, is focused on the quality of the top predictions and is appropriate for recommendation tasks. For each sampled representative entity we compute the entities with top k cosine similarity and consider the average fraction of that set that are in the same block.

#blocks		0.75		0.95		0.99	
B	peak	%gain	$\times 10^9$	%gain	$\times 10^9$	%gain	$\times 10^9$
Cosine Gap Quality							
10	1.09	31.07	1.58	24.80	1.89	**19.06**	2.17
20	1.03	29.56	1.40	23.74	1.72	**18.13**	1.97
50	1.00	25.52	1.22	20.46	1.55	**14.73**	1.80
100	0.99	20.66	1.09	15.27	1.40	**12.11**	1.64
Precision at $k = 10$ Quality							
10	1.00	43.64	1.12	37.70	1.24	**37.41**	1.38
20	1.00	40.80	1.03	36.16	1.14	**33.47**	1.23
50	1.00	33.89	0.92	31.37	1.04	**25.35**	1.09
100	1.00	26.67	0.84	23.37	0.94	**20.00**	1.02

Table 1. Training gain of COO with respect to IND baseline for stochastic blocks ($d = 50$, $b = 64$). Peak is maximum quality for COO. We report the training for IND to reach 75% , 95%, and 99% of peak with respective percent reduction in training with COO.

6.4. Stochastic blocks results

Our results were consistent for different dimensions and minibatch sizes and we report representative results for $d = 50$ and $b = 64$ and for the methods COO, IND, and the reference method COO+OPTLSH. Results for the cosine gap quality measure and for the precision (at $k = 10$) are reported in Figure 3 and Table 1. The figures show the increase in quality in the course of training for the different methods . The x-axis in these plots shows the *amount of training* in terms of the *total number of gradient updates performed*. The tables report the amount of additional training needed for IND to obtain the performance of COO. We observe that across all block sizes B and for the two quality measures, COO arrangement resulted in significantly faster convergence than the IND arrangements. The gains were larger with larger blocks. Much of the gain of COO arrangements over IND was realized earlier in training and then maintained. The COO+OPTLSH method provided only very modest improvement over COO and only for larger blocks. This improvement bounds that possible by any COO+LSH method on this data and indeed COO+LSH results (not shown) were between COO and COO+OPTLSH.

6.5. Recommendation data sets and results

We performed experiments on two recommendation data sets, MOVIELENS1M and AMAZON. The MOVIELENS1M dataset (Movielen1M) contains 10^6 reviews by 6×10^3 users of 4×10^3 movies. The AMAZON dataset (SNAP) contains 5×10^5 fine food reviews of 2.5×10^5 users on 7.5×10^3 food items. Provided review scores were [1-5] and we preprocessed the matrix by taking κ_{ij} to be 1 for review score that is at least 3 and 0 otherwise. We then reweighed entries in the MOVIELENS1M dataset by dividing the value by the sum of its row and column to the power of 0.75. This is standard processing that retains only positive ratings and reweighs to prevent domination of frequent entities.

We created a test set T_+ of positive examples by sampling 20% of the non zero entries with probabilities proportional to κ_{ij}. The remaining examples were used for training. As

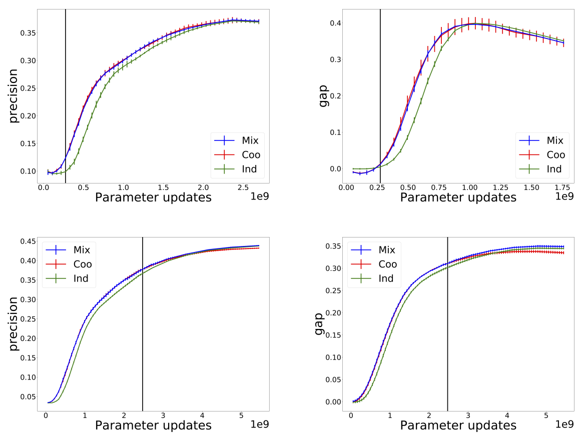

Figure 4. Precision at $k = 10$ (left) and cosine gap (right) in the course of training with different arrangement methods on MOVIELENS1M(top) and AMAZON(bottom) datasets. ($d = 50$, $b = 64$). The vertical lines indicate the switch point of MIX (from COO to IND).

	$0.75\times$ peak		$0.95\times$ peak		$0.99\times$ peak	
	%gain	$\times 10^9$	%gain	$\times 10^9$	%gain	$\times 10^9$
AMAZON cosine gap: Gain over IND (peak=0.35)						
Mix@4.5M	9.8	1.56	10.29	3.12	**12.55**	3.94
Coo	9.48	1.56	7.35	3.12	0	3.94
MOVIELENS1M cosine gap: Gain over IND (peak=0.40)						
Mix@0.25M	11.94	0.68	7.45	0.82	**6.08**	0.92
Coo	11.94	0.68	4.97	0.82	**6.08**	0.92
AMAZON precision: Gain over IND (peak=0.44)						
Mix@4.5M	10.76	1.80	3.00	3.40	**4.05**	4.53
MOVIELENS1M precision: Gain over IND (peak=0.37)						
Mix@0.25M	10.24	0.85	4.31	1.66	**3.24**	2.05

Table 2. AMAZON and MOVIELENS1M, cosine gap and precision, training gain over IND baseline ($b = 64$, $d = 50$).

negative test examples T_{-} we used random zero entries. We measured quality using the cosine gap and precision at $k = 10$ over users with at least 20 nonzero entries. We used 5 random splits of the data to test and training sets and 10 runs per split. The results are reported in Figure 4 and Table 2. We show performance for COO and IND arrangements and also for a MIX method that started out with COO arrangements and switched to IND arrangements at a point determined by a hyperparameter search. The MIX method was often the best performer. We observe consistent gains of 3%-12% that indicate that arrangement tuning is an effective tool also on these more complex real-life data sets.

7. Example Selection Experiments

In this section we empirically explore the qualities of the information contained in fractions of epochs when using COO and IND arrangements. We select a small set of training examples that corresponds to a fractional epochs with different arrangements and then train to convergence using

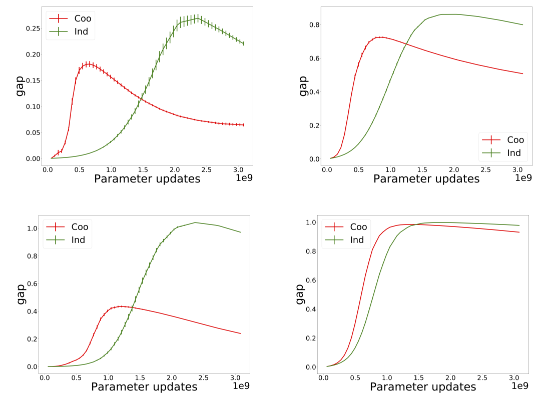

Figure 5. Training with selected examples (fractional epoch with IND and COO arrangements). Stochastic blocks datasets and minibatch size parameter $b = 4$. Left to right: $(T = 5, B = 10)$; $(T = 5, B = 100)$, $(T = 20, B = 10)$, $(T = 20, B = 100)$.

multiple epochs on only the selected examples. We sampled $T = 5, 10, 15, 20$ examples from each row (for focus updates) and symmetrically from each column (for context updates) of the association matrix. To emulate a fractional IND arrangement, we select T independent examples from each row i by selecting a column j with probability $\kappa_{ij}/\|\kappa_{i\cdot}\|_1$. To emulate a fractional COO arrangement we repeat the following T times. We draw $u_j \sim \text{Exp}[1]$ for each column and select for each row i the column $\arg\max_j \kappa_{ij}/u_j$. Clearly the marginal distribution of both selections is the same: The probability that column j is selected for row i is equal to $\kappa_{ij}/\|\kappa_{i\cdot}\|_1$. Symmetric schemes apply to columns.

We trained embeddings (with no bias parameter and using IND arrangements) on these small subsets of examples using identical setups. Updates were according to minibatch designation: Row samples used for updating row embeddings and column samples for updating column embeddings. Representative results are reported in Figure 5. We observe that COO selection consistently results in faster early training than IND selection but sometimes reaches a lower peak. We explain the faster early training by the COO selection preserving short-range similarities (based on first-hop relations) and the lower peak by lost "long-range" structure (that reflects longer-range relations as those captured by longer random walks and metrics such as personalized page rank). Our COO arrangements which use the complete set of examples retain both the short-range benefits of COO selections and the long-range benefits of IND selections.

8. Conclusion

We demonstrated that SGD can be accelerated with principled arrangements of training examples that are mindful of the "information flow" through gradient updates.

Acknowledgements

We are grateful to the anonymous ICML 2019 reviewers for many helpful comments that allowed us to improve the presentation. This research is partially supported by the Israel Science Foundation (Grant No. 1841/14).

References

Abadi, M. and et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Alain, G., Lamb, A., Sankar, C., Courville, A. C., and Bengio, Y. Variance reduction in SGD by distributed importance sampling. *CoRR*, abs/1511.06481, 2015. URL http://arxiv.org/abs/1511.06481.

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *ICML*, 2009.

Berry, M. W., Dumais, S. T., and O'Brien, G. W. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595, 1995.

Brewer, K. R. W., Early, L. J., and Joyce, S. F. Selecting several samples from a single population. *Australian Journal of Statistics*, 14(3):231–239, 1972.

Broder, A. Z. Identifying and filtering near-duplicate documents. In *Proc.of the 11th Annual Symposium on Combinatorial Pattern Matching*, volume 1848 of *LNCS*, pp. 1–10. Springer, 2000.

Bromley, J., Guyon, I., Lecun, Y., Säckinger, E., and Shah, R. Signature verification using a "siamese" time delay neural network. In *In NIPS Proc*, 1994.

Chopra, S., Hadsell, R., and LeCun, Y. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, 2005.

Cohen, E. Size-estimation framework with applications to transitive closure and reachability. *J. Comput. System Sci.*, 55:441–453, 1997.

Cohen, E. Distance queries from sampled data: Accurate and efficient. In *KDD*. ACM, 2014. full version: http://arxiv.org/abs/1203.4903.

Cohen, E., Kaplan, H., and Sen, S. Coordinated weighted sampling for estimating aggregates over multiple weight assignments. *VLDB*, 2, 2009. full version: http://arxiv.org/abs/0906.4560.

Cohen, G. Auxiliary problem principle and decomposition of optimization problem. *J Optim Theory Appl*, 32, 1980.

Condon, A. and Karp, R. M. Algorithms for graph partitioning on the planted partition model. *Random Struct. Algorithms*, 18(2), 2001.

Csiszar, I. and Tusnády, G. Information geometry and alternating minimization procedures. *Statistics & Decisions: Supplement Issues*, 1, 1984.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. Indexing by latent semantic indexing. *Journal of the American Society for Information Science*, 41(6):391–407, September 1990.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Li, F.-F. Imagenet: A large-scale hierarchical image database. In *In CVPR*, 2009.

Duchi, J., Hazan, E., and Singer, Y. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12, 2011. URL http://dl.acm.org/citation.cfm?id=1953048.2021068.

Dumais, S. T. Latent semantic indexing (LSI): TREC-3 report. In Harman, D. K. (ed.), *The Third Text Retrieval Conference (TREC-3)*, pp. 219–230, Gaithersburg, MD, 1995. U. S. Dept. of Commerce, National Institute of Standards and Technology.

Fu, T. and Zhang, Z. Cpsg-mcmc: Clustering-based preprocessing method for stochastic gradient mcmc. In *AISTATS*, 2017.

Gemulla, R., Nijkamp, E., Haas, P. J., and Sismanis, Y. Large-scale matrix factorization with distributed stochastic gradient descent. In *ACM KDD 2011*. ACM, 2011.

Goemans, M. X. and Williamson, D. P. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42 (6), 1995.

Hu, Y., Koren, Y., and Volinsky, C. Collaborative filtering for implicit feedback datasets. In *ICDM*, 2008.

Kish, L. and Scott, A. Retaining units after changing strata and probabilities. *Journal of the American Statistical Association*, 66(335):pp. 461–470, 1971. URL http://www.jstor.org/stable/2283509.

Koren, Y. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *KDD*, 2008.

Koren, Y., Bell, R., and Volinsky, C. Matrix factorization techniques for recommender systems. *Computer*, 42, 2009.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

Movielen1M. Movielen 1M Dataset. URL `http://grouplens.org/datasets/movielens/1m/`.

Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *EMNLP*, 2014.

Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *KDD*. ACM, 2014.

Robbins, H. and Siegmund, D. O. A convergence theorem for non negative almost supermartingales and some applications. In Rustagi, J. S. (ed.), *Optimizing Methods in Statistics*. Academic Press, 1971.

Salakhutdinov, R., Mnih, A., and Hinton, G. Restricted boltzmann machines for collaborative filtering. In *ICML*. ACM, 2007.

Salton, G. and Buckley, C. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5):513 – 523, 1988.

Schroff, F., Kalenichenko, D., and Philbin, J. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2015.

SNAP. Stanford network analysis project. `http://snap.stanford.edu`.

word2vec.py. tensorflow models. URL `https://github.com/tensorflow/models/blob/master/tutorials/embedding/word2vec.py`.

Zhang, C., Kjellström, H., and Mandt, S. Stochastic learning on imbalanced data: Determinantal point processes for mini-batch diversification. In *UAI 2017*, 2017.

Zhao, P. and Zhang, T. Stochastic optimization with importance sampling for regularized loss minimization. In *ICML*, 2015.